Some Generalizations of Bucket Brigade Assembly Lines

A Thesis
Presented to
The Academic Faculty

by

Yun Fong Lim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Industrial and Systems Engineering

Georgia Institute of Technology
April 08, 2005

Some Generalizations of Bucket Brigade Assembly Lines

Approved by:

Dr. John J. Bartholdi, III, Chair, Co-Adviser
School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Donald D. Eisenstein, Co-Adviser
Graduate School of Business
University of Chicago

Dr. Robert D. Foley
School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. Leon F. McGinnis
School of Industrial & Systems Engineering
Georgia Institute of Technology

Dr. John H. Vande Vate
School of Industrial & Systems Engineering
Georgia Institute of Technology

Date Approved: April 08, 2005

To my parents

Acknowledgments

I thank Professor John Bartholdi for his guidance and financial support. Without him the accomplishment of this thesis would have been impossible. Special thanks must go to Professor Donald Eisenstein, who always inspires me on my work, for co-advising me. I thank Professors Robert Foley, Leon McGinnis, and John Vande Vate for giving many useful, interesting comments on the draft of this thesis.

I would also like to express my gratitude to Yetkin Ileri, Melda Ormeci, Feryal Kuran, Deniz Dogan, Devang Dave, Stephanie Jernigan, Brady Hunsaker, Paul Brooks, Eda Kemahlioğlu Ziya, Yongpei Guan, Na An, Jinxiang Gu, Tiravat Assavapokee, Sriram Subramanian, James Luedtke, I-Lin Wang, Wen-Chih Chen, Yen-Tai Wan, Yaxian Zhang, Gary Ng, and Hwa Liang Leo for their company and encouragement along the way. I am sure you guys will come into my mind many years later when I recall the days in Atlanta.

I thank my parents for giving me the opportunities to pursue my own interests. They are always confident, optimistic, and supportive on my decisions. Last but not least, I would like to thank my wife, Pei Ching, for her love, patience, and constant support in many ways throughout these years.

Contents

List of Tables

List of Figures

Summary

A fascinating feature of bucket brigade assembly lines is that work load on workers is balanced spontaneously as workers follow some simple rules in the assembly process. This self-organizing property significantly reduces the management effort on an assembly line. We generalize this idea in several directions. These include an adapted bucket brigade protocol for complex assembly networks, a generalized model that permits chaotic behavior, and a more detailed model for a flow line in which jobs arrive arbitrarily in time and are introduced into the system at several points on the line. We report both analytical and simulation results.

Chapter 1

Bucket Brigades

Bucket brigades are used in industry as a way of sharing work among workers on assembly lines. They were originally used in apparel manufacturing for the assembly of sewn products [10], and were later found to be extremely effective in order-picking in warehouses [9, 12]. One of the most attractive properties of bucket brigades is that the work allocation to workers on an assembly line is balanced spontaneously as workers follow some simple rules in the assembly process. This *self-balancing* feature significantly reduces the management effort on the assembly line. A selection of case studies on the use of bucket brigades in industry can be found in [9].

1.1 The Normative Model

Consider an assembly line such as shown in Figure 1.1, in which a product is progressively assembled on a sequence of work stations. When workers form a bucket brigade, every

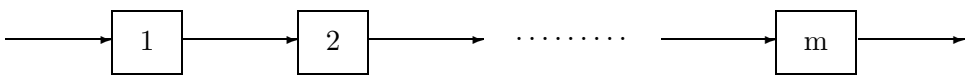

Figure 1.1: An assembly line in which each instance of the product is progressively assembled on the same sequence of work stations

Table 1.1: The Bucket Brigade Rules determine what each individual team member should do.

Forward Rule: Work forward with your item until

1. your item is handed off to your successor; or
2. you complete your item if you are the last worker on the line;

then follow the Backward Rule.

Backward Rule: Walk back to get more work,

1. if you are the first worker on the line, begin a new item at the start of the line;
2. otherwise, take over the item from your predecessor;

then follow the Forward Rule.

worker simultaneously carries an instance of the product (called an *item*) along the assembly line. In the Normative Model of bucket brigades [10] workers are not allowed to pass one another so that their sequence along the assembly line is preserved. When the last worker finishes his work on an item at the end of the line, he walks back upstream and takes over the work from his predecessor, who in turn walks back and receives the work from *his* predecessor, and so on, until the first worker initiates a new item at the beginning of the line. Each worker in a bucket brigade follows two simple rules stated in Table 1.1.

The Normative Model of bucket brigades relies on the following assumptions.

Assumption 1.1. (Smoothness and Predictability of Work)

The work content of the product is a constant (which we normalize to 1); and it is distributed continuously and uniformly over the assembly line. Furthermore, the work content at any work station is preëmptible without significant loss of work.

Figure 1.2 shows a conceptual flow line in which the total work content of the product is represented as a line segment normalized to length 1. Each item is initiated at the location 0 (start of the line) and is completed at the location 1 (end of the line). The location of worker i is denoted as x_i, which represents the cumulative fraction of work

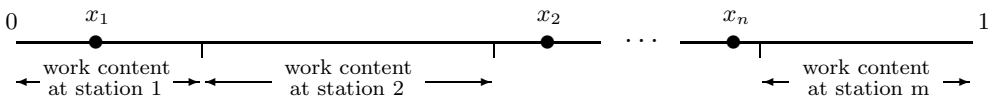

Figure 1.2: The total work content of the product is distributed continuously and uniformly over the assembly line and it is conceptually represented by a line segment normalized to length 1. This line segment is partitioned into intervals corresponding to the work stations. The location of worker i is denoted as x_i, which represents the cumulative fraction of work content completed on his item.

content completed on his item. The state of the system at any instant in time can be represented by the vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where $0 \leq x_1 \leq x_2 \leq \ldots \leq x_n \leq 1$.

Assumption 1.2. (Workers Are Characterized by Work Velocities)

Each worker i, $1 \leq i \leq n$, is characterized by a distinct, constant work velocity v_i.

The work velocity of a worker reflects his familiarity with the work content of the product.

Assumption 1.3. (Insignificant Walking Time)

The time to walk the length of the entire assembly line is negligible compared to the total time to assemble an item. Therefore, workers walk back with an infinite velocity and all hand-offs occur simultaneously.

All hand-offs occur *at the same instant*, immediately after the last worker finishes an item at the end of the line. We call such an instant a *reset*. Let $\mathbf{x}^k$ denote the locations of workers immediately after the kth reset. Note that $x_1^k = 0$. We can study the dynamics of the system by focusing on the sequence $\{\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k, \ldots\}$. This simplifies the analysis as we do not need to trace the details of the continuous-time evolution of the system.

1.2 Self-balance

A bucket brigade assembly line is *balanced* if the following two conditions are satisfied:

Repetition: Each worker repeats the same portion of work content on each successive instance of the product.

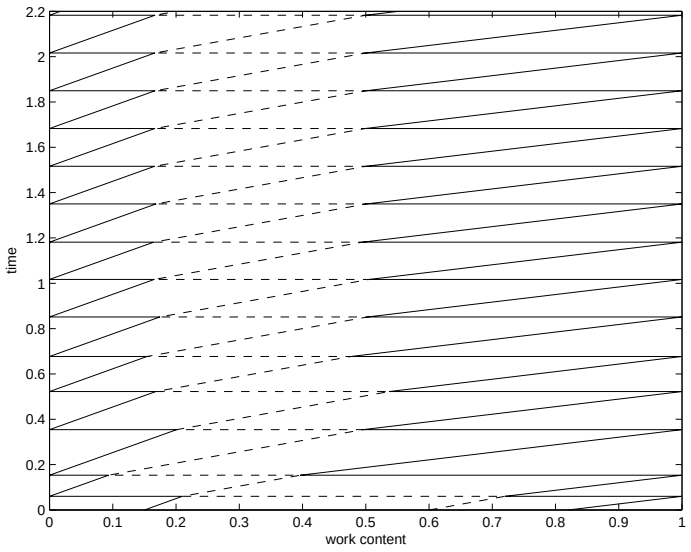

Figure 1.3: **The evolution of a three-worker bucket brigade assembly line:** The horizontal axis represents the total work content of the product and the vertical axis represents time. Workers are sequenced from slowest to fastest in the direction of production flow and they start from arbitrary initial locations on the assembly line. The zigzag vertical lines show how the locations of workers change over time and each of the rightmost spikes corresponds to a completed item. The system quickly converges to a stationary state on which each worker covers a fixed portion of work content on each item produced.

Efficiency: Workers are utilized to their fullest.

Balance is desirable because then the skills of each worker are reinforced by repetition, the effort of each worker is directly realized as output of finished product, and production is regular, which simplifies downstream processes.

It was shown in [10] that if workers are sequenced from slowest to fastest (according to their work velocities) in the direction of production flow, that is, $v_1 < v_2 < \ldots < v_n$, then the system converges to a *fixed point* [2, 19] at which every worker repeats the same portion of work content on each item produced. Mathematically, $\mathbf{x}^k \to \mathbf{x}^*$ as $k \to \infty$, where

$$x_i^* \quad = \quad \frac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j}, \text{ for } i = 1, \ldots, n. \tag{1.1}$$

Figure 1.3 shows the evolution of the locations of workers on a bucket brigade assembly line as time progresses. In this example, there are three workers with work velocities equal

to 1, 2, and 3. Workers are sequenced from slowest to fastest in the direction of production flow and they start from arbitrary initial locations on the assembly line. As shown in the figure, the system converges to a fixed point upon which each worker repeatedly works on the same portion of work content on each item produced.

Furthermore, upon the fixed point the throughput (number of items produced per unit time) of the line attains $\sum_{i=1}^{n} v_i$, the maximum possible. Thus, the throughput can be adjusted by simply adding or removing workers from the line as long as workers are sequenced from slowest to fastest.

The bucket brigade assembly line is said to *self-balance* when workers are sequenced from slowest to fastest in the direction of production flow. This *self-organizing* property of bucket brigades is particularly appealing because the system achieves balance without the intervention of management.

1.3 Dynamics of two- and three-worker lines

Bartholdi *et al.* [8] analyzed the long-run (asymptotic) dynamics of the Normative Model with two and three workers. For two-worker lines they found only two modes of asymptotic behavior available: If workers are sequenced from slowest to fastest along the line in the direction of production flow the system converges to a fixed point of optimal throughput; otherwise a *2-cycle* (also known as *period-2 orbit*, see [2, 19] for an introduction) is found. The 2-cycle has suboptimal throughput, twice the work velocity of the slower worker.

For three-worker lines they found more complex dynamics. Depending on the work velocities of workers, the system either converges to a fixed point of optimal throughput or a *k-cycle* ($k \geq 2$) of suboptimal throughput. In particular, in a parameter space defined by the relative work velocities of workers, they found a region with $k > 3$ in which the asymptotic behavior may depend not only on the velocities but also on the initial locations of workers. The rich dynamical behavior of the system suggests wariness in interpreting some simulation results of bucket brigades in [15], [31], and [32], in which processing times

are assumed to be random. It can be hard to distinguish the effects of "real" randomness (in processing times) on the observed behavior from that of the complex dynamics of the system.

1.4 When are bucket brigades appropriate?

In contrast to traditional assembly line balancing (see, for example, [14, 30]), which adopts rigid work assignment to workers, workers in a bucket brigade are not restricted to any particular zone of the assembly line or assignment of work content. This allows the system to dynamically and spontaneously reallocate work by moving workers to where the work is. The use of bucket brigade assembly seems most appropriate when:

- *All work on the assembly line is based on a single skill.* This allows workers to work at any stations along the line so that they can share their work by moving to where the work is. It also allows workers to be ranked by a single score (that is, their work velocity along the line), so that workers can be sequenced from slowest to fastest.

- *The turnover rate of workers is high.* New and inexperienced workers are often introduced into a team as a result of the departure of some workers. A new worker is less familiar with the work content than the existing, experienced workers in the team. Thus, workers can be distinguished by their work velocities.

- *Workers can move easily among stations and can easily take over work from co-workers.* This ensures that workers do not spend additional, wasted time to take over work.

- *Work content of the product varies significantly.* The assignment of work to workers need not be carefully balanced because a bucket brigade constantly seeks balance from item to item. It will be out of balance much less often compared to a rigid work assignment.

1.5 Recent developments

The Normative Model has been generalized or extended to deal with more complex situations. These include when the work content is stochastic, when workers have varying velocities along the assembly line, when workers spend significant time to walk back, and when the work velocities change due to learning. Furthermore, perhaps more interestingly, the idea of bucket brigades can be used to interpret some behavior of social insects.

1.5.1 Bucket brigades with stochastic work content

Bartholdi *et al.* [12] analyzed a bucket brigade assembly line with stochastic work content. They assumed that the processing times at work stations are independent and follow an exponential distribution. They showed analytically that, as the number of stations increases, the dynamics of the stochastic model increasingly resembles that of the normative model (which attains optimal throughput when workers are sequenced from slowest to fastest). Thus, the bucket brigade protocol remains effective in the presence of variability of processing time when there is sufficient work distributed among sufficiently many stations. They validated the stochastic model in the order-picking operation at a distribution center of a major chain retailer and confirmed the effectiveness of bucket brigades [12].

1.5.2 When workers cannot be sequenced from slowest to fastest

What should we do when workers on an assembly line cannot be sequenced from slowest to fastest? Can a bucket brigade achieve self-balancing? Armbruster and Gel [5] analyzed the dynamics and throughput of two-worker bucket brigades when the work velocities of workers do not dominate each other uniformly along the assembly line. One worker, say worker B, has a constant velocity along the entire line. The other worker, say worker A, works slower than B in the first segment of the line and faster in the second segment. This model is useful when workers possess different skill levels on different parts of the line. They assumed that the work content is deterministic and it is distributed continuously

and uniformly over the line. They considered two different modeling assumptions when the upstream worker catches up with the downstream worker:

1. The upstream worker can continue his work by passing the downstream worker.

2. The upstream worker is blocked by the downstream worker and proceeds with the work velocity of the latter.

Under the first modeling assumption, the system either converges to a fixed point or a 2-cycle if we switch the order of workers on the line when the upstream worker passes his co-worker and finishes the work on his item at the end of the line. Thus, the asymptotic dynamics of the two-worker bucket brigade cannot be more complex than a 2-cycle. Similar results were found under the second modeling assumption. Armbruster and Gel [5] also discussed the trade-off between the throughput of the system and the benefit brought by self-balancing.

1.5.3 When the walk-back time is significant

Bratcu and Dolgui [16] studied a relaxation of the Normative Model by assuming every worker walks back with *the same* finite, constant velocity. They found that if workers are sequenced from slowest to fastest according to their work (forward) velocities in the direction of production flow, then the bucket brigade will converge to a fixed point on which each worker repeatedly executes the same portion of work content on each item produced. The expression of the fixed point is different from Equations (1.1) in the way that it incorporates the effects of the walk-back velocity. They established a relation between the new fixed point and Equations (1.1).

1.5.4 Bucket brigades with finite walk-back velocities and passing

In some situations workers in a bucket brigade assembly line are allowed to pass each other and they may spend significant time to walk back for more work. Bartholdi and Eisenstein

[11] introduced a generalized model in which workers can pass each other (thus, they are not restricted to a fixed sequence along the line). Furthermore, each worker walks back with a finite velocity to receive work. They found a new, generalized condition for a bucket brigade to converge to a fixed point.

In addition, they observed *chaotic* behavior on this deterministic model through simulations when the above condition is violated. This interesting result suggests that the variability in the output of the assembly line due to the dynamics of the system can be very large. We analyze the chaotic dynamics of this model in detail in Chapter 4.

1.5.5 Bucket brigades with labor turnover and learning

In environments with high labor turnover, such as the maquiladora industry in Northern Mexico [23], new and inexperienced workers are introduced into assembly lines as a result of the departure of experienced workers. Since new workers are inexperienced the time to complete a product becomes longer and more variable. This results in serious degradation on the performance of the assembly line [23].

Muñoz and Villalobos [27] investigated the potential of the bucket brigade protocol in environments with high labor turnover. They used a simple model to capture labor turnover and the learning process of new workers. Furthermore, they introduced several worker replacement policies, which determine the sequence of workers along a line when a new worker is introduced into the line as a result of the departure of an experienced worker. Results show that when the turnover rate is significant, bucket brigades equipped with an appropriate worker replacement policy outperform the traditional balanced lines (in which a static work load is allocated to each worker). The self-balancing feature allows bucket brigades to better absorb variability caused by labor turnover. Similar ideas are pursued in [26], where processing times on tasks are assumed to be exponentially distributed.

Armbruster *et al.* [6] studied the dynamics and throughput of bucket brigades when the velocities of workers increase due to learning. They considered a bucket brigade with

experienced workers sequenced from slowest to fastest. To this bucket brigade, they added a new, inexperienced worker. Since the new worker learns only those parts of the assembly line that he works on, the asymptotic distribution of his velocity is non-uniform over the line. In the case in which workers are allowed to pass each other when one catches up with another, a bucket brigade will typically reorganize the workers so that the system converges to a fixed point. On the other hand, if passing is not allowed then a proper initial placement of the new worker is crucial for the bucket brigade to converge to a fixed point. They also briefly discussed the situation in which all workers are initially new and improve their velocities through learning.

1.5.6 Bucket brigades in nature

Do bucket brigades occur in nature? Reyes and Fernández-Haegar [29] observed that *Messor barbarus* ants carry seeds from the food source to the nest using a transport scheme that is consistent with bucket brigades. These ants transfer seeds directly from one to another along a trail from the food source to the nest. Transfers do not take place at predetermined locations but occur where an ant meets another on the trail. Interestingly, seeds are successively transferred from smaller ants to larger ants. The smallest ant forages out farthest from the nest. When carrying a seed back toward the nest, she may be interrupted by a larger ant, who wrests the seed from her and continues carrying it toward the nest. After the largest ant relinquishes her seed at the nest, she goes back out to get another.

Anderson *et al.* [3] provide an explanation on how bucket brigades arise in this context. They assumed that larger ants move faster and an ant can take a seed from a smaller ant but not from a larger one. Under these assumptions a bucket brigade emerges spontaneously as each ant follows a simple rule. See [3] for more bucket brigades in insect societies.

1.6 What can we find in this thesis?

We generalize the ideas of bucket brigades to more complex environments. Our work consists of the following three parts. Each part deals with different complications encountered in practice.

Bucket brigades on in-tree assembly networks

In a network of subassembly lines, balance becomes more difficult to achieve as it requires the system to satisfy the following three conditions:

Repetition: Each worker repeats the same portion of work content on each successive instance of the product.

Efficiency: Workers are utilized to their fullest, while the work-in-process remains bounded.

Synchronization: All subassembly lines produce at a common rate.

We focus on networks with in-tree structure in which different subcomponents are produced on different subassembly lines and completed subcomponents are then successively assembled to produce the final product. We show how to adapt the bucket brigade protocol of work-sharing so that balance emerges spontaneously on such networks in Chapter 2. The idea developed in Chapter 2 is also applicable to a more restrictive situation in which workers must travel along the subassembly lines and they are not allowed to pass each other. Chapter 3 describes how bucket brigades can be run on these more restrictive assembly networks.

Stability and chaos on bucket brigades

In Chapter 4 we discuss the generalized model introduced in [11] in which workers can pass each other (thus, they are not restricted to a fixed sequence along the line) and each worker walks back with a finite velocity to receive work. This model is more appropriate

in some situations. For instance, in some low-density order-picking operations workers are allowed to pass each other and they must walk considerable distance between picks. The time to work forward to pick a few stock-keeping-units is not significantly larger than the time to walk back upstream for more work.

Even for the case with only two workers, the dynamics of this model may be very complicated. More specifically, when the system satisfies a certain condition it converges to a fixed point at which workers always hand off items at a fixed location and no passing occurs. Thus, each worker repeats a portion of work content on each successive item and each item is completed in a fixed interval of time. On the other hand, if the condition is violated the system behaves *chaotically*. Hand-offs do not converge to a fixed location. Instead, the hand-off locations spread erratically along the assembly line as items are assembled. Workers pass each other as they assemble their items and they do not concentrate on a fixed portion of work content. Furthermore, this chaotic behavior induces large variability in completion times. Results show that the variability on the system could be so large that the output of this fully deterministic system is effectively random if the system is not properly configured.

Bucket brigades with multiple job sources: A case study

In Chapter 5 we adapt the bucket brigade protocol for a flow line on which jobs arrive arbitrarily in time and they are introduced into the system at multiple points on the line. This work is motivated by the order-picking operation of a distribution center at Atlanta, where customer orders arrive arbitrarily in time and they are released through different printers at different locations on an aisle. We evaluate the adapted protocol through computer simulations based on real data from the distribution center. In particular, we analyze the labor and time spent on order-picking and consolidation. We compare the performance of the adapted protocol under different configurations with zone-picking, which is a policy currently adopted by the distribution center.

Chapter 2

Bucket Brigades on In-Tree Assembly Networks

In a *network* of subassembly lines, balance becomes more difficult to achieve as it requires that all subassembly lines be synchronized to produce at the same rate. We show how to adapt the bucket brigade protocol of work-sharing so that balance emerges spontaneously.

2.1 In-tree assembly networks

Consider a network of subassembly lines such as shown in Figure 2.1, where each arc represents a subassembly line that produces a subcomponent (or, equivalently, a part). The subassembly lines merge until a last line produces the final product, each instance of which we call an *item*. We restrict consideration to assembly networks that are *in-trees*: Each node has exactly one leaving arc, except for the root node (node K in Figure 2.1), which has none. The flow of assembly is *in* toward a final assembly line from which finished product emerges.

Arc j is that one emanating from node j and it represents the work content on subassembly line j, on which subcomponent j is assembled. Each subassembly line j has a buffer j located at the end of the line to store completed instances of subcomponent j.

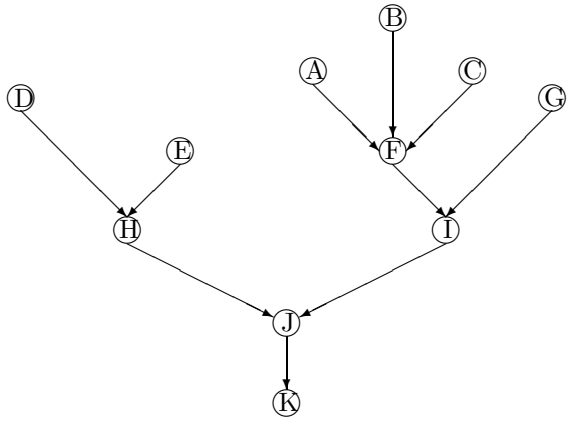

Figure 2.1: An assembly network is represented by a connected directed graph, which represents the constraints on the sequence in which subcomponents are assembled.

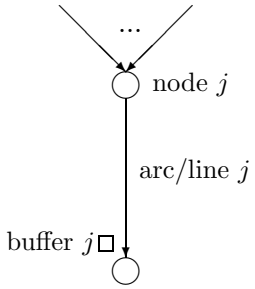

Figure 2.2: Subcomponent j is assembled on line j and deposited in buffer j.

Figure 2.2 shows the relationship between node j, line j, and buffer j.

A *leaf node* is a node with no entering arcs on the in-tree, such as A, B, and C in Figure 2.1. Assembly of different subcomponents is initiated at leaf nodes, and these subcomponents are assembled in moving along the subassembly lines devoted to them. At an interior node such as F in Figure 2.1, subcomponents are joined to the assembly of a larger subcomponent F. Each instance of the finished product is completed and leaves the network at the root node, in this example K.

We assume that the work-content on each subassembly line has already been determined and is described by the following.

Assumption 2.1. *The work to process each subcomponent is deterministic and is spread*

continuously and uniformly over the corresponding subassembly line. Furthermore the work-content at any station is preëmptible without significant loss of work.

An example of such a line is one with relatively many work stations, each of which has a small piece of the total work-content. As discussed in previous work, these assumptions are not strictly necessary but they greatly simplify analysis [8, 10, 12].

Let there be n workers on the assembly in-tree. Our model of them is intended to capture simple, unskilled assembly in which workers are fully cross-trained:

Assumption 2.2. *Each worker $i = 1, \ldots, n$ is characterized by a work velocity v_i that is fixed and constant over all subassembly lines.*

Assume distinct workers labeled so that $v_1 < v_2 < \ldots < v_n$.

2.2 Balance

An in-tree assembly network is *balanced* if the following three conditions are satisfied:

Repetition: Each worker repeats the same portion of work content on each successive instance of the product.

Efficiency: Workers are utilized to their fullest, while the work-in-process remains bounded.

Synchronization: All subassembly lines produce at a common rate.

Balance is desirable because then the skills of each worker are reinforced by repetition, the effort of each worker is directly realized as output of finished product, and production is regular, which simplifies downstream processes.

Traditional assembly-line balancing forces repetition by assigning workers to zones. It tries to achieve efficiency by computing perfect shares of work-content, accounting for differences in the velocities of the workers. (For example, see [14, 30], though these ignore,

as does almost the entire modern literature, the significant differences among worker velocities.) Synchronization may be sought by trying to make work-shares similar across all subassembly lines.

Unfortunately, this approach requires the construction of a detailed model of work-content and knowledge of the exact velocities of the workers. Even if balance is achieved on average (even this is hard), it would be difficult to maintain balance under the vicissitudes of the factory floor. Consequently, as a practical matter, synchronization is achieved, if at all, by sacrificing efficiency and accepting waste in the form of either some idleness among workers or else in growing work-in-process, especially between subassembly lines.

Bartholdi and Eisenstein [10] proposed the bucket brigade protocol to balance linear assembly lines (that is, assembly lines that may be represented as single arcs). In contrast to traditional assembly line balancing, workers on a linear assembly line running the bucket brigade protocol are not restricted to any particular zone of the line or assignment of work-content. This allows the system to dynamically reallocate work by moving workers to where the work is. If workers are sequenced from slowest to fastest (according to their work velocities) along the direction of material flow, the system spontaneously converges to a stationary state in which every worker repeatedly executes the same portion of work on every item produced [10]. We say the line balances itself. The system is also efficient — the number of units of work-in-process never exceeds the number of workers and the throughput (number of products finished per unit time) is the maximum possible [10].

Up to now, bucket brigades have been defined and studied only on linear assembly lines. The goal of this chapter is to explore how the bucket brigade protocol can be extended to make an in-tree of subassembly lines self-balancing. To use the same idea on an in-tree we must specify exactly how the workers should move among the subassembly lines.

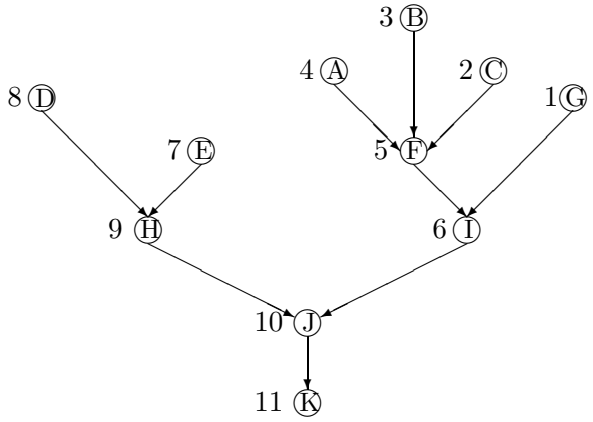

Figure 2.3: A total ordering of the subassembly lines: Each node has been assigned an ordinal number representing the order of the subassembly line emanating from that node. Each worker follows this sequence of work when he proceeds forward in the assembly process. Note that this total order of nodes is consistent with the partial order of subassembly lines defined by the in-tree.

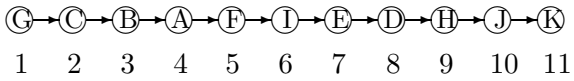

Figure 2.4: The serialized assembly line is a conceptual representation of a total order of work on the in-tree of Figure 2.3.

2.3 What should a worker do next?

We implicitly determine what workers do next (that is, where to move) by conceptually transforming an in-tree of subassembly lines into a single linear assembly line that represents a common sequence of work-content to be completed by the workers in the course of production. Later we will identify some new complications this introduces into the familiar form of bucket brigades and show how to adapt bucket brigades to handle these complications.

Consider an in-tree of subassembly lines, such as the one illustrated in Figure 2.1. We assume that the physical layout of the assembly network is fixed (we are not laying out the

Table 2.1: The Bucket Brigade Rules determine what each individual team member should do. This version of the rules accounts for a new phenomenon, the possibility of "starvation".

Forward:	Assemble your item moving from one subassembly line to the next according to the serialized ordering. If you are the last worker and complete an item or if your item is taken over by your successor then go **Back**. If you are *starved* — that is, if a required subcomponent is unavailable — then drop your work in the buffer at the end of the subassembly line you just completed and go **Back**.
Back:	If you are the first worker then return to the first subassembly line according to the serialized ordering and start a new item and go **Forward**; otherwise walk back to your predecessor, take over his item, and go **Forward**.

assembly line). To adapt the bucket brigade protocol to an in-tree structure, we impose a linear sequence on the work-content of the in-tree. This linear sequence is chosen to be consistent with the partial ordering imposed by the in-tree itself; since the sequence is a total ordering, we can treat it as a linear assembly line to which (some form of) bucket brigades might be applied.

Figure 2.3 shows one total order of the subassembly lines from among the many that are consistent with the partial order of lines of the in-tree. For example, line D precedes line H in any total order because processing a subcomponent H requires a subcomponent D. The total order of work is illustrated in Figure 2.4, which is a conceptual representation of the sequence in which the work will be completed. We refer to a total ordering like the one shown in Figure 2.4 as a *serialized assembly line*.

Having totally-ordered all the work-content of the in-tree, the main points of the bucket brigade protocol are now well-defined. In particular, both the initialization (sequence workers from slowest to fastest) and the basic rule of movement (work "forward" and walk "back" to get more work) are understood in relation to this total order.

Thus, under bucket brigades for in-trees the partial order conveyed by the in-tree is serialized, and each worker follows the rules given in Table 2.1.

In Figure 2.4 the slowest worker in a team is always the first worker we encounter when proceeding from node G. As in the standard bucket-brigade protocol, when the last worker (the fastest worker) finishes his work at node K, he walks back upstream and takes over the work of his predecessor (the next slower worker), who will be located at some point on a subassembly line. This slower worker in turn walks back and takes over the work of *his* predecessor, and so on, until the first worker begins a new subcomponent at node G.

2.4 Two new issues

This adaptation of the bucket brigade protocol introduces two new issues that do not arise in simple linear assembly lines. First is the issue of travel. When following the bucket brigade protocol on the serialized assembly line, workers must travel from one subassembly line to another and these subassembly lines may be physically far apart. It is clear that the larger the distances between subassembly lines, the greater the potential for wasted productive capacity as workers travel between lines. For now we shall assume that this travel is not "too large"; in Section 2.6.2 we shall show how to reduce such travel.

The total time to complete an instance of the product produced by the in-tree assembly network under the extended bucket brigade protocol is the sum of *productive time* and *unproductive time*. The productive time is the time spent exclusively in the assembly of the product, while the unproductive time is that spent traveling but not assembling (either forward to the next subassembly line or backward to get more work). For now we adopt the following assumption.

Assumption 2.3. *The unproductive time is negligible compared to the productive time.*

The second new issue is that the forward progress of a worker on the serialized assembly line can be halted if, on starting work on a subassembly line, he finds that a required subcomponent is not available (the corresponding buffer is empty). For example, when a worker finishes a subcomponent D, according to the sequence of work shown in Figure 2.4,

he should proceed to line H. However, he might not be able to continue his work if buffer E is empty. We say the worker is *starved* and we extended the original bucket brigade protocol to handle this situation. The possibility of starvation raises the concern that worker capacity could be systematically diverted to the assembly of subcomponents without proportional completions of the final product. Is the bucket brigade system stable? Can buffers grow without bound?

2.5 Starvation is transient

With respect to any total ordering, the item currently being assembled by worker i partitions the subassembly lines of the in-tree into two sets: the set $L_i^<$ of subassembly lines whose completion points (and buffers) occur prior to the current position of the worker; and the set $L_i^\geq$, the current and remaining lines on the in-tree.

Definition 2.1. *The item being assembled by worker i is* fully provisioned *if the following condition holds: each of the buffers of the subassembly lines in $L_i^<$ that are required for assembly in $L_i^\geq$ holds a subcomponent uniquely reserved for the assembly of that item. A bucket brigade is fully provisioned if each of the items being assembled is fully provisioned.*

This definition can be checked worker-by-worker, examining the buffers in each case and labeling the contents. This requires $O(mn)$ steps, where m is the number of subassembly lines. The importance of this idea lies in the following, which is a direct result of the definition.

Observation 2.1. *If the bucket brigade is fully provisioned then the item represented by each worker will be completed without starvation among the workers.*

The bucket brigade protocol works to establish a natural association between subcomponents in the buffers and instances of the product as follows: Imagine that, when a new item is begun, it is assigned a unique, fixed identification number and this number is revealed to each worker when they take over production of the item. Imagine further that

each worker labels any subcomponent he assembles with the identification number of the item for which he is currently responsible and thus any necessary subcomponents retrieved for the item also share the unique identifier.

Lemma 2.1. *Each new item that enters the bucket brigade system is fully provisioned and remains so as it progresses to completion.*

Proof. Since the serialized line observes the precedence constraints, each required subcomponent is assembled before it is needed and left in its buffer. □

Corollary 2.1. *If a bucket brigade is fully provisioned then it will remain fully provisioned.*

Proof. This follows because the natural association persists between items in the hands of workers and the subcomponents that were assembled for them. □

Lemma 2.2. *There can be no more than n instances of starving.*

Proof. By the extended bucket brigade protocol stated in Table 2.1, each instance of starving triggers a walk-back, which starts a new instance of the product. After n walk-backs all n items in the system (in the hands of bucket brigade workers) are fully provisioned and so there can be no more starving. □

Theorem 2.1. *If workers are sequenced from slowest to fastest in the direction of the serialized assembly line then a bucket brigade on an in-tree will balance itself.*

Proof. Lemmata 2.1 and 2.2 ensure that a bucket brigade on an in-tree will quickly achieve a fully provisioned state regardless of the initial state of the system, hence any starvation is transient. Corollary 2.1 guarantees that the bucket brigade, upon reaching a fully provisioned state, will remain in a fully provisioned state. Thus due to Assumption 2.3, a bucket brigade operating in a fully provisioned state on an in-tree eventually behaves as a traditional bucket brigade on a linear assembly line independent of its initial state (see Theorem 3 of [10]). □

The following tells us that, under bucket brigades, the buffers cannot grow without bound. This will be important to establish the throughput of the system. Let m be the number of subassembly lines.

Lemma 2.3. *From any initial conditions (positions of the workers, state of buffers) the total number of subcomponents in buffers can never grow by more than mn before the bucket brigade is fully provisioned. After it is fully provisioned, the total population of subcomponents cannot grow beyond an additional mn.*

Proof. By the extended bucket brigade protocol, each instance of starvation triggers a walk-back, in which the starved worker in effect abandons the item on which he was working and goes back to take over the item of his predecessor. This leaves in the buffers no more than m subcomponents intended for the abandoned item. By Lemma 2.2 there can be at most n instances of starvation, during which the population of the buffers cannot grow more than mn.

After starving has ceased each new subcomponent in a buffer corresponds to a unique instance of the product; and at any time that instance of the product is carried by a unique worker who is downstream of the buffer. But there are only n workers and m buffers, so the total population of the buffers cannot grow by more than mn. □

Theorem 2.2. *If unproductive time is negligible in comparison to work-content, the bucket brigade line on an in-tree spontaneously balances itself, so that eventually each worker repeats the same interval of work-content and all subassembly lines produce at a common rate $\sum_i v_i$, which is the largest possible.*

Proof. From Assumptions 2.1 and 2.3 workers are always productive, and since by Lemma 2.3 the amount in the buffers is finite, eventually all productive capacity will be realized as output. By Theorem 3 of [10], each worker repeats an interval of work-content at a common cycle time and so each subassembly line produces at the common rate $\sum_i v_i$. □

By Theorem 2.2 the long-run dynamics of bucket brigades on assembly in-trees are similar to those for linear assembly lines. Nevertheless, it is worth remarking that the short-term dynamics can differ in that bucket brigades on assembly in-trees are susceptible to a new form of disruption: If, in the midst of production, a subcomponent is found to be flawed and unsuitable for use, bucket brigades can proceed through a sequence of n instances of starvation before reëstablishing a fully provisioned state (Lemma 2.2), and, in the process, can generate spurious subcomponents. For example, consider M subassembly lines meeting at a common node at which final assembly begins. Assume the M lines are sequenced $1, 2, \ldots, M$ and imagine a single worker assembling each of these subcomponents in turn and depositing it in the appropriate buffer. If the subcomponent in buffer 1 is later found to be defective, the worker will not be able to proceed to final assembly, but will instead, under the extended bucket brigade protocol, retrace his path and then assemble another of each of the subcomponents $1, 2, \ldots, M$. The bucket brigade will have abandoned the item for which the flawed subcomponent was intended, along with all other subcomponents intended for it, and gone blindly back to begin construction of another item. In short, rather than try to recover in the fastest possible way, bucket brigades maintain simplicity and consistency of movement. Nevertheless, Lemma 2.2 guarantees that bucket brigades will recover quickly and Lemma 2.3 guarantees that not "too many" spurious subcomponents will be produced during recovery.

2.6 What is the best total order?

There are many ways to realize a total order of subassembly lines that is consistent with the partial order of these lines on the in-tree. Here, we discuss two criteria by which one total order may be preferred to another. The first criterion, which we call *Progress Toward Subgoals*, favors total orders in which partially completed subcomponents are moved along as far as possible before new subcomponents are introduced. The second criterion is *Unproductive Time*, which is a concern if travel between subassembly lines are significant

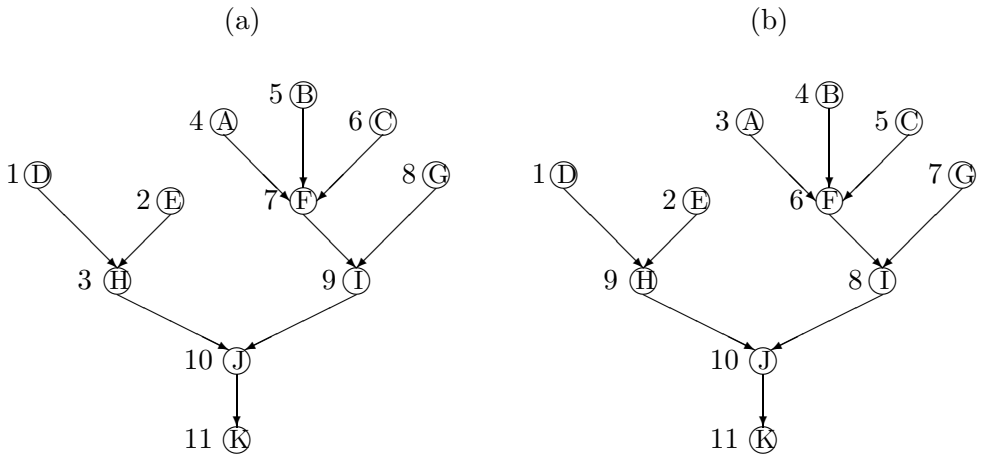

Figure 2.5: (a) A total order of nodes that is in the preferred set: The work sequence constructed in this order maintains progress toward subgoals in the assembly process. (b) A total order of nodes that is not in the preferred set: This order does not maintain progress toward subgoals because a worker finishing his work on line E will proceed to line A instead of line H. This violates the condition of continuity of work flow.

(that is, Assumption 2.3 fails to hold). Total orders that minimize travel distance are preferred.

It is possible to deal with either of these issues independently — maintaining progress toward subgoals or reducing unproductive time — however, we choose to account for them both in the following way: we will select, from among the total orders that maintain progress toward subgoals, one that requires minimal unproductive time (that is, one that requires minimal travel).

2.6.1 Progress toward subgoals

We favor total orders in which each subcomponent is completed as much as possible before new subcomponents are begun. To illustrate this criterion, suppose there is a worker working on line D in Figure 2.1 and assume that all the buffers on the in-tree are empty. When the worker finishes his work on line D, he may begin a new subcomponent at any of nodes E, A, B, C, or G. Taking the continuity of work flow into account, it is preferable for him to begin to work on line E and then line H before proceeding to another subtree. The intuition is to complete the current subcomponent before processing other

less-directly-related subcomponents. This may be formalized in the following way: *Before assembling subcomponent j, recursively assemble all subcomponents that immediately precede j.* The reader will realize that this can be achieved by running a typical *reverse search algorithm* (see, for example, [1] and [18]). We enforce this in the belief that it helps workers understand the meaning behind their work and so learn it faster and execute it more reliably.

We define the *preferred set* as the set of total orders that maintain progress toward subgoals in the assembly process. According to this definition, the total orders of nodes shown in Figures 2.3 and 2.5(a) are in the preferred set. Figure 2.5(b) gives an example that is not in the preferred set because a worker finishing a subcomponent E will begin a subcomponent A instead of completing the work on the left subtree rooted at node J.

2.6.2 Minimizing travel by workers

Types of travel

Following the extended bucket brigade protocol, workers on the in-tree assembly network can either travel forward (moving in the direction consistent with the total order of lines to assemble products) or travel backward (moving in the reverse direction to get more work). More specifically, we may classify all travel as follows.

Forward travel, which can be classified into two types:

- **Productive travel,** in which workers advance the assembly of the product as they proceed along the subassembly lines.
- **Unproductive travel,** when a worker finishes one subcomponent and travels (without working) to the start of the next subassembly line.

Backward travel, in which a worker walks back to take over the work of his predecessor. Backward travel is always unproductive.

In addition, we make the following assumption.

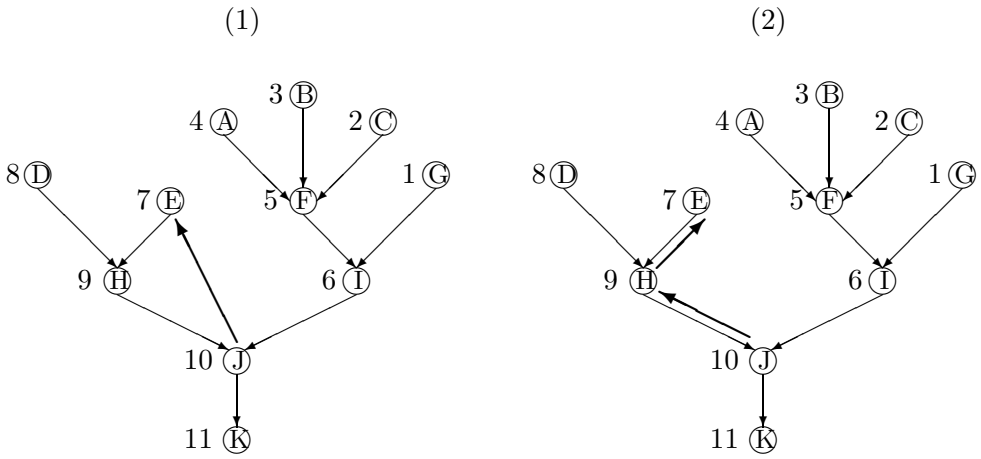

Figure 2.6: There are different ways for a worker to travel between subassembly lines. A worker finishing subcomponent I (line 6) travels to start subcomponent E (line 7) via two alternative ways: (1) he follows the most direct path between nodes J and E; (2) he walks along the shortest path between nodes J and E on the in-tree (that is, $J \to H \to E$). When walking back, he retraces the path chosen for forward travel.

Assumption 2.4. *When traveling backward, a worker retraces the forward path.*

This assumption is conservative, in that it ignores opportunities to take short-cuts when walking back to get work from a predecessor. However, in practice such opportunities are often not available — potential short-cuts may be hampered by conveyors or shelving that limit movement between lines. Furthermore, a worker must *find* his predecessor to get more work, so that retracing the forward path may be necessary simply to preserve the bucket brigade protocol.

Workers must travel, both forward and backward, between subassembly lines and this travel is unproductive. Since we want to minimize (unproductive) travel we must be able to measure the travel between subassembly lines. There are two reasonable models for travel between subassembly lines.

1. Travel is along the most direct path between two nodes (the end of a subassembly line and the start of another).

2. Travel is along the shortest path between the two nodes *on the in-tree*.

Figure 2.6 illustrates these two alternatives for travel between subassembly lines. Regardless of which model of travel is adopted, we assume the following is true.

Assumption 2.5. *The distance of travel from one subassembly line to another is known.*

Due to Assumption 2.4, the total forward and backward travel required to complete each item is independent of the number of workers on the line. Consequently it is sufficient to study a 1-worker system.

Observation 2.2. *The problem of finding a total order of subassembly lines for which the travel is minimized for an n-worker system is equivalent to the same problem for a 1-worker system.*

Due to Observation 2.2, and for the sake of simplicity, we assume that there is only one worker in the system in the following discussion on finding a total order of lines with minimal travel. Furthermore, when walking back to get more work workers traverse, in the reverse direction, the same path they follow in forward travel. Thus, it is sufficient to consider only forward travel.

Pattern of travel

Define the *joint node* as the node where all subcomponents meet to produce the final product. A joint node represents the beginning of the final subassembly line and it is unique on an in-tree. In Figure 2.3 node J is the joint node. Consider a worker working on the in-tree according to some order in the preferred set. For illustration, assume that he follows the order shown in Figure 2.3. He begins his work from line 1 (line G in Figure 2.3), which is on some subtree rooted at the joint node. Let T be the subtree rooted at the joint node that contains line 1. In Figure 2.3 the subtree T contains lines A, B, C, F, G, and I. After finishing the work on line 1 the worker then subsequently traverses all the other lines on T, following the order of lines, until he reaches the joint node. According to the order in Figure 2.3, he traverses the lines in the sequence $G \to C \to B \to A \to F \to I$.

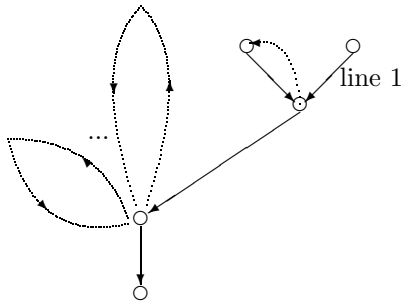

Figure 2.7: A schematic plot of the pattern of forward travel: A worker traverses the first subtree rooted at the joint node through a preferred walk. He then subsequently traverses each of the remaining subtrees through a preferred tour. He finally completes an instance of the product on the final subassembly line.

Upon reaching the joint node, the worker continues his work on another subtree rooted at the joint node in a similar manner. Note that there are only two subtrees rooted at the joint node in Figure 2.3 but, in general, there could be more than two subtrees rooted at the joint node on the in-tree. The worker visits a sequence of subtrees in a consecutive manner. Every time the worker finishes traversing all the lines on a subtree, he returns to the joint node where he begins his traversal on the next subtree, and so on. Finally, he proceeds to line J, which is the final subassembly line on the in-tree. Note that this pattern of motion is consistent with any order that maintains progress toward subgoals.

Let s be a subtree rooted at the joint node on an in-tree. Let A_s be the set of arcs on s, where each arc is represented by an ordered pair of nodes. Define a *walk* on s as a sequence of nodes $(j_1, j_2, \ldots, j_m)$ satisfying the constraint that for each arc $(u, v) \in A_s$ there is some q such that $j_q = u$ and $j_{q+1} = v$. A walk may visit a node on s more than one time. A walk that starts from a leaf node on s, ends at the joint node, and maintains progress toward subgoals is called a *preferred walk*. On the other hand, a walk that starts and ends at the joint node and maintains progress toward subgoals is called a *preferred tour*.

When the worker proceeds forward in assembling the product, following the pattern of travel described above, he first traverses all the arcs on the first subtree T through a preferred walk. He then traverses each of the remaining subtrees rooted at the joint node through a preferred tour. Finally, he completes an instance of the product on the final subassembly line. Figure 2.7 illustrates the pattern of forward travel.

Procedure to find total order with minimal travel

Given an in-tree such as shown in Figure 2.1, we want to find a total order of lines that minimizes the total unproductive travel in the assembly process subject to the constraint that the order must maintain progress toward subgoals. In other words, our problem is to find a total order from the preferred set such that the unproductive travel in producing an instance of the product is minimized. This can be achieved by determining:

- the first subtree T rooted at the joint node (or equivalently, choosing line 1) and a preferred walk that traverses T; and
- a preferred tour on each of the remaining subtrees rooted at the joint node;

such that the sum of lengths of the preferred walk on T and all the preferred tours is minimized.

Let $\mathcal{S}$ be the set of subtrees rooted at the joint node. We find a total order with minimal travel from the preferred set using the following procedure.

1. For each subtree $s \in \mathcal{S}$:
   - find a preferred walk w_s on s with minimum length C_s^w;
   - find a preferred tour t_s on s with minimum length C_s^t.
2. Let the subtree $T = \text{argmax}_{s \in \mathcal{S}}\{C_s^t - C_s^w\}$.
3. Construct the partial order of lines on T according to the order of lines that w_T traverses T.

4. For all $s \in \mathcal{S}\backslash\{T\}$, construct the partial order of lines on s according to the order of lines that t_s traverses s.

5. Construct the total order of all the lines on the in-tree by combining the partial orders obtained in steps 3 and 4. Traverse the subtree T first, and then traverse the remaining subtrees in $\mathcal{S}$ in an arbitrary order.

In step 1 we need to solve two problems:

- finding a minimum preferred walk, and
- finding a minimum preferred tour,

for each subtree $s \in \mathcal{S}$. The first problem is a special instance of the second when the distance from the joint node to any leaf node on the in-tree is set to 0.

Heuristic to find minimum preferred tour

Interestingly, the problem of finding a minimum preferred tour on a subtree, rooted at the joint node of an in-tree, can be modeled as the stacker crane problem with precedence constraints. In the original, unconstrained stacker crane problem [20], we are given a set $\mathcal{N}$ of nodes (together with the corresponding distance matrix) and a set $\mathcal{A}$ of arcs, where each arc is an ordered pair of nodes. The objective is to search for a tour $(j_1, j_2, \ldots, j_m)$, where $j_1 = j_m$, with minimum length subject to the constraint that for each arc $(u, v) \in \mathcal{A}$, there is some q such that $j_q = u$ and $j_{q+1} = v$. Note that a tour may visit a node (other than node j_1) in $\mathcal{N}$ more than one time. The stacker crane problem with precedence constraints is the stacker crane problem with additional constraints requiring that certain arcs must be visited before others. In our problem these precedence constraints represent the precedence relations between the subassembly lines on the in-tree as well as the constraint that the partial order of lines must maintain progress toward subgoals.

The stacker crane problem is NP-complete [20]. Frederickson *et al.* [20] suggested two polynomial time approximation algorithms to solve the stacker crane problem. In

particular, the first algorithm (called the *large arcs* algorithm) is suitable for the case when the total length of arcs in $\mathcal{A}$ is large compared to the optimal tour length. It returns a solution with cost no greater than $3C^* - 2C_{\mathcal{A}}$, where C^* and $C_{\mathcal{A}}$ represent the length of the optimal tour and the total length of arcs in $\mathcal{A}$ respectively. Furthermore, the algorithm terminates in $O(|\mathcal{A}|^3)$ time. Interested readers can refer to [20] for the details.

The large arcs algorithm can be adapted to solve the stacker crane problem with precedence constraints described above. The adaptation is straightforward. It simply removes some forbidden links between arcs in $\mathcal{A}$ to capture the precedence constraints. The adapted algorithm has the same worst-case performance guarantee and time complexity.

Although the problem of finding a minimum preferred tour on a subtree is NP-complete, we should note that all real in-tree assembly networks with which we are familiar are fairly simple and would not require actual application of the algorithmic procedure described above to generate practical solutions.

Special case

When the travel from one subassembly line to another follows the shortest path between two nodes on the in-tree (that is, alternative 2 in Figure 2.6), the problem of finding a minimum preferred tour on a subtree, rooted at the joint node of the in-tree, can be solved to optimality in polynomial time. In this more restricted case, all preferred tours on a subtree rooted at the joint node have the same length. Thus, we only need to search for a partial order of lines that is consistent with the precedence relations between the subassembly lines and maintains progress toward subgoals. This can be achieved by running a typical *reverse search algorithm* (see, for example, [1] and [18]) on the subtree. Furthermore, the problem of finding a minimum preferred walk on the subtree can be solved to optimality in polynomial time in a similar manner.

2.7 Related work

In 1994 Gershwin observed that "there has been relatively little written on Assembly/ Disassembly networks compared with the vast queuing theory literature" [21]. We find the situation unchanged today. The literature of which we are aware models the assembly network as a queuing network, typically with iid exponential processing times (for example, [21]). Typically the emphasis is on describing steady state behavior of the system, if any, and measuring work-in-process and throughput.

Some previous work seeks balance by the assignment of work content. Most relevant to our work is that of Baker *et al.*, who considered an assembly system in which two subcomponents are produced at different stations and then joined at third station [7]. The processing times at all stations are stochastic and follow the same distribution. No buffers are used in the system, but the assembly station can accept completed subcomponents while awaiting other subcomponents.

Baker *et al.* focused on the cases in which the total work-content is constant and is equal to the number of stations in the system and the coefficient of variation at each station is fixed. For this model Baker *et al.* investigated how to allocate the total work-content (sum of mean processing times at all stations) to the stations so that the throughput of the system is maximized. They found that the assembly system should be "unbalanced" in the direction of assigning less work to the assembly station and more to component stations. They also considered a generalized system in which two feeder lines merge at an assembly station. Each feeder line consists of an initial work station and a final work station, where the latter feeds the assembly station. Their results indicate that throughput is maximized by allocating decreasing amounts of work-content closer to assembly.

In contradistinction, we have assumed that the work content has already been allocated among the work stations, but that — as to be expected in the real world — the resulting balance will be imperfect, both moment-to-moment due to stochastic variance and in the long-run, due to impossibility of dividing the work perfectly. Bucket brigades can smooth

over those imperfections to achieve, and dynamically maintain, near perfect balance.

2.8 Summary

The main contribution of our work is to show how the idea of bucket brigades [10] can be applied to in-tree assembly networks, so that even relatively complex assembly systems can enjoy the benefits of self-balancing. We achieve this by conceptually converting the work-content on an assembly network to a one-dimensional sequence of work upon which the adapted bucket brigade protocol can be executed. The system converges to a stationary state in which every worker repeats the same portion of work-content (possibly across several subassembly lines) on each successive instance of the product. Furthermore, to a good approximation, the throughput of the system attains the maximum possible.

It is worth remarking that some managerial issues that are trivial on simple, linear bucket brigades may present practical problems on complicated in-trees. For example, workers must follow a common sequence of work, which means that the sequence of subassembly lines must be either memorized or else marked clearly for both forward and backward movement. Similarly, on walking back to get more work, workers must be able to find and recognize their predecessors.

If travel time is significant then of course the throughput of the assembly system would be diminished because workers would spend more time in unproductive travel and less in actual assembly. In addition, significant travel times could disrupt the self-balancing, and in particular the tendency of workers to repeat the same interval of work content. This could happen if, for example, the in-tree is such that when a worker walks back from position x the time to travel back is very different from the time required from position $x + \epsilon$.

In practice it may be quite easy to implement bucket brigades on in-tree assembly systems because real assembly trees are fairly simple, at least in our experience. For example, a motivating application for our work was an assembly in-tree to produce large

screen televisions for Mitsubishi [9]. It was a simple "Y" shape and so the appropriate total ordering was obvious.

Chapter 3

Bucket Brigades on More Restrictive In-Trees

3.1 Assembly in-trees that are more restrictive

The approach in Chapter 2 assumes that workers on an in-tree assembly network can travel from one node to another either by

1. the most direct path between the nodes; or
2. the shortest path between the two nodes on the in-tree.

In the latter case, workers are allowed to pass each other when they travel along the subassembly lines on the in-tree. However, in some situations workers *must* travel along the subassembly lines on the in-tree as direct travel from one node to another may not be available due to shelving, conveyors, etc. Furthermore, when workers travel along the subassembly lines they may not be allowed to pass each other due to space limitation (for example, narrow aisles) or some technical constraints that forbid passing between workers.

Here, we demonstrate ways of running a bucket brigade on an in-tree when workers must travel along the subassembly lines and are not allowed to pass each other. As the

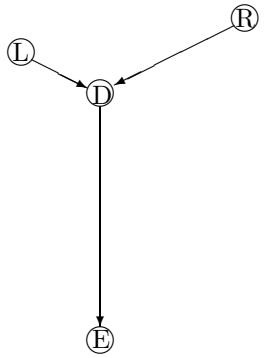

Figure 3.1: A simple assembly in-tree consists of only three subassembly lines.

motion of workers becomes more restrictive, it becomes more difficult to make bucket brigades self-balanced on an in-tree. Part of the difficulty is that workers can no longer travel freely to where the work is and care must be taken to prevent starvation. Due to the complexity of the analysis we focus on a two-worker bucket brigade on a simple "Y" shape in-tree.

3.2 Simple assembly in-trees

Consider a simple assembly in-tree shown in Figure 3.1. It consists of only three arcs, each of which corresponds to one subassembly line. Arc j is that one emanating from node j and it represents the work content on subassembly line j, on which subcomponent j is assembled. Each subassembly line j has a buffer j located at the end of the line to store completed instances of subcomponent j.

Assembly of subcomponent L and subcomponent R are initiated at nodes L and R respectively. These subcomponents are assembled in moving along the subassembly lines devoted to them. Subcomponents are joined at node D to the assembly of final product. Each instance of the product, called an *item*, is completed and leaves the in-tree at the root node, in this example E.

We assume that the work content on each subassembly line has already been determined

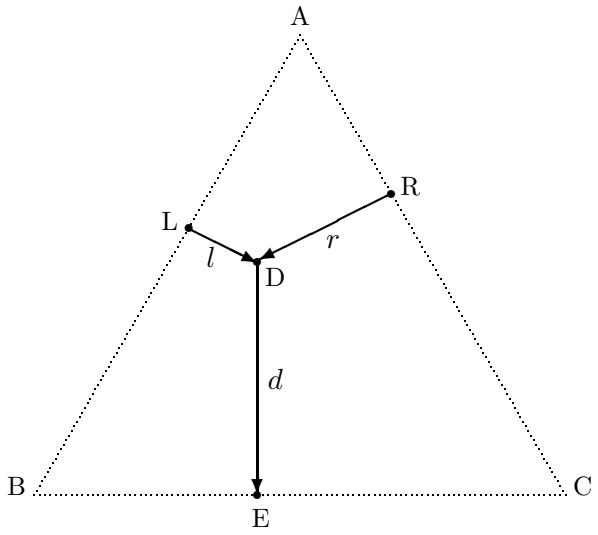

Figure 3.2: The simple assembly in-tree is embedded in an equilateral triangle ABC. Subassembly lines L, R, and D are perpendicular to the edges AB, AC, and BC respectively. The sum of the lengths l, r, and d is a constant for node D located arbitrarily within the triangle. For the sake of simplicity the total work on the system is normalized to 1, so that $l + r + d = 1$.

and is described by the following.

Assumption 3.1. *The work to process each subcomponent is deterministic and is spread continuously and uniformly over the corresponding subassembly line. Furthermore the work content at any station is preëmptible without significant loss of work.*

An example of such a line is one with relatively many work stations, each of which has a small piece of the total work content.

Thus, the amount of work on subassembly lines L, R, and D can be represented by lengths l, r, and d respectively. To facilitate our analysis, we utilize a geometrical property of equilateral triangles. Assume that the entire structure of the in-tree is embedded in an equilateral triangle ABC as depicted in Figure 3.2. Subassembly lines L, R, and D are perpendicular to the edges AB, AC, and BC respectively. It can be shown easily that the sum of the lengths l, r, and d is a constant for node D located arbitrarily within the triangle. For the sake of simplicity the total work on the system is normalized to 1, so that $l + r + d = 1$.

We consider the case where there are only two workers in the system. Our model of them is intended to capture simple, unskilled assembly in which workers are fully cross-trained:

Assumption 3.2. *Each worker $i = 1, 2$ is characterized by a work velocity v_i that is fixed and constant over all subassembly lines.*

Assume distinct workers labeled so that $v_1 < v_2$. Furthermore, we consider more restrictive situations as stated in the following assumption.

Assumption 3.3. *Workers must travel along the subassembly lines on the in-tree and they are not allowed to pass each other.*

The total time to complete an instance of the product produced by the simple assembly in-tree is the sum of *productive time* and *unproductive time*. The productive time is the time spent exclusively in the assembly of the product, while the unproductive time is that spent traveling but not assembling (either forward to the next subassembly line or backward to get more work). For now we adopt the following assumption.

Assumption 3.4. *The unproductive time is negligible compared to the productive time.*

3.3 Balance

We adopt the same definition of balance on an in-tree stated in Chapter 2. A simple assembly in-tree is *balanced* if the following three conditions are satisfied:

Repetition: Each worker repeats the same portion of work content on each successive instance of the product.

Efficiency: Workers are utilized to their fullest, while the work-in-process remains bounded.

Synchronization: All subassembly lines produce at a common rate.

Table 3.1: The Bucket Brigade Rules determine what each individual team member should do.

Forward: Assemble your item moving from one subassembly line to the next according to the serialized ordering. If you are the last worker and complete an item or if your item is taken over by your successor then go **Back**.
Back: If you are the first worker then return to the first subassembly line according to the serialized ordering and start a new item and go **Forward**; otherwise walk back to your predecessor, take over his item, and go **Forward**.

Balance is desirable because then the skills of each worker are reinforced by repetition, the effort of each worker is directly realized as output of finished product, and production is regular, which simplifies downstream processes.

3.4 What should a worker do next?

We adopt the same approach introduced in Chapter 2 to make the simple assembly in-tree balanced: Serialize the subassembly lines so that each worker conforms to the same sequence of work during the assembly of the product. However, due to Assumption 3.3, not every total order of subassembly lines that satisfies the partial ordering imposed by the in-tree is appropriate. We will find out the correct total order of subassembly lines in the next section. For now, we assume that this correct total order is given so that some form of bucket brigades can be constructed. In particular, both the initialization (sequence workers from slowest to fastest) and the basic rule of movement (work "forward" and walk "back" to get more work) are understood in relation to this total order.

After the subassembly lines on the simple in-tree are serialized into a correct total order, workers are sequenced from slowest to fastest with respect to this total order. Thus, according to this total order worker 1 always begins an item from the first subassembly line and worker 2 always completes an item on the final subassembly line. Each worker follows the rules given in Table 3.1.

Note that the rules given in Table 3.1 do not specify what to do if a worker is starved (that is, if a required subcomponent is unavailable). Instructions for workers in case of starvation are unnecessary because starvation will never occur if the subassembly lines are serialized into a correct order and workers are initialized at appropriate locations on the in-tree. Thus, workers on the in-tree do not start from arbitrary locations. Some proper initial locations must be chosen with respect to the order of subassembly lines to prevent starvation.

3.5 What is the correct total order?

We focus on the case in which point D is located within the left portion of the triangle in Figure 3.2, that is $l \leq r$. The analysis for the case $l > r$ is similar. Define the parameter

$$s \equiv \frac{v_1}{v_1 + v_2}$$

to represent the relative work velocities of workers. Note that $0 < s < 1/2$. Depending on the values of s, l, and r, the correct total order of subassembly lines can be different. Define the following four disjoint regions:

***Region 1*:** $s < l$;

***Region 2*:** $l < s < r$;

***Region 3*:** $r < s < l + r$;

***Region 4*:** $s > l + r$.

Each region has a different way of serializing the subassembly lines to make the two-worker bucket brigade on the simple in-tree balanced.

Let x^k denote the location of the kth hand-off on the simple in-tree in Figure 3.2. We construct the reference frame such that $x^k \in [0, l]$ on line L, $x^k \in (l, l + r]$ on line R, and $x^k \in (l + r, 1]$ on line D. At each iteration k, worker 1 hands item k off to worker 2 at

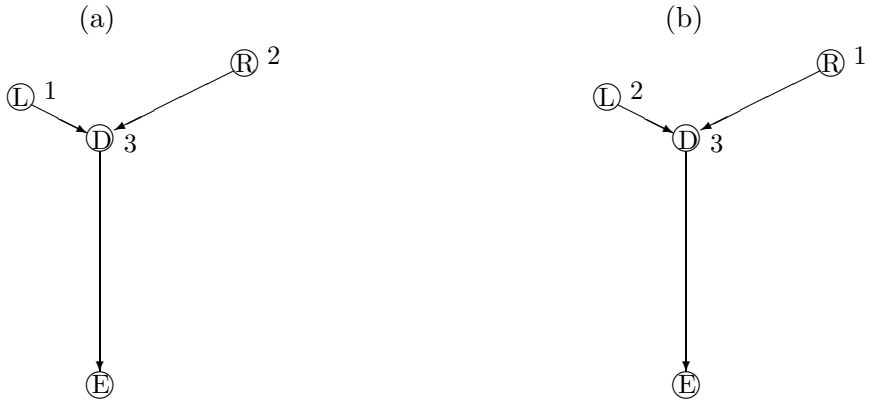

Figure 3.3: (a) Subassembly lines are sequenced as $L \to R \to D$. (b) Subassembly lines are sequenced as $R \to L \to D$.

location x^k. Worker 1 returns to the first subassembly line (according to the total order of lines) to begin item $k+1$ while worker 2 continues to work on item k. When worker 2 completes item k, he walks back to receive item $k+1$ from worker 1 at x^{k+1}.

3.5.1 *Region 1* $(s < l)$

There are two different ways to serialize the subassembly lines to make the two-worker bucket brigade balanced. Figure 3.3 shows two different orders of subassembly lines. In Figure 3.3 (a) the lines are sequenced as $L \to R \to D$. Lemma 3.1 shows the existence and uniqueness of a fixed point on the in-tree under this ordering of subassembly lines.

Lemma 3.1. *If the subassembly lines are sequenced as $L \to R \to D$ in Region 1, then there exists a unique fixed point on line L and it is given by*

$$x_L^* = \frac{v_1}{v_1 + v_2}. \tag{3.1}$$

Proof. Following the order of lines $L \to R \to D$, the fixed point x_L^* is found by solving

$$\frac{x_L^*}{v_1} = \frac{1 - x_L^*}{v_2}.$$

Since $x_L^* = s < l$, the fixed point is located on line L. □

When the system operates on the fixed point x_L^* each worker repeatedly executes a fixed portion of work content of each instance of the product: After worker 1 hands item k off to worker 2 at x_L^*, worker 1 returns to node L to begin item $k+1$ while worker 2 continues to work on item k. When worker 2 finishes the work on line L for item k he proceeds to line R and then line D. Finally, when worker 2 completes item k at node E he returns to x_L^* to receive item $k+1$ from worker 1. Note that workers do not pass each other when they travel along the subassembly lines.

Lemma 3.2. *If the subassembly lines are sequenced as $L \rightarrow R \rightarrow D$ in Region 1, and $x^0 \in [max(0, 1-(v_2/v_1)l), l]$, then $x^k \in [max(0, 1-(v_2/v_1)l), l]$, for $k = 1, 2, 3, \ldots$.*

Proof. We first prove that if $x^{k-1} \in [max(0, 1-(v_2/v_1)l), l]$, then $x^k \in [max(0, 1-(v_2/v_1)l), l]$. Following the order of lines $L \rightarrow R \rightarrow D$, we have

$$\begin{aligned} \frac{x^k}{v_1} &= \frac{1-x^{k-1}}{v_2}; \\ x^k &= v_1\left(\frac{1-x^{k-1}}{v_2}\right). \end{aligned}$$

Case 1. If $1-(v_2/v_1)l \geq 0$, $x^{k-1} \in [1-(v_2/v_1)l, l]$ implies

$$\begin{aligned} x^k &\leq v_1\left[\frac{1-[1-(v_2/v_1)l]}{v_2}\right]; \\ &= l; \end{aligned}$$

and

$$\begin{aligned} x^k &\geq v_1\left(\frac{1-l}{v_2}\right); \\ &> 1-\frac{v_2}{v_1}\, l. \end{aligned}$$

The last inequality is due to the fact that $s < l$. Thus, $x^k \in [1-(v_2/v_1)l, l]$.

Case 2. If $1-(v_2/v_1)l < 0 \Rightarrow v_1/v_2 < l$, $x^{k-1} \in [0, l]$ implies

$$\begin{aligned} x^k &\leq v_1\left(\frac{1}{v_2}\right); \\ &< l; \end{aligned}$$

and

$$\begin{aligned} x^k &\geq v_1\left(\frac{1-l}{v_2}\right); \\ &> 0. \end{aligned}$$

Thus, $x^k \in [0, l]$. Since k is arbitrary we conclude that if $x^0 \in [max(0, 1-(v_2/v_1)l), l]$, then $x^k \in [max(0, 1-(v_2/v_1)l), l]$, for $k = 1, 2, 3, \ldots$. □

Lemma 3.2 shows that if $x^0 \in [max(0, 1-(v_2/v_1)l), l]$ then the hand-off locations x^k are always on line L.

Lemma 3.3. *If the subassembly lines are sequenced as $L \rightarrow R \rightarrow D$ in Region 1, and $x^0 \in [max(0, 1-(v_2/v_1)l), l]$, then the two-worker bucket brigade converges to the fixed point x_L^*.*

Proof. According to Lemma 3.2, if $x^0 \in [max(0, 1-(v_2/v_1)l), l]$ then the hand-off locations x^k are always on line L. Let $x^k = x_L^* + \delta^k$. Since

$$\begin{aligned} \frac{x^k}{v_1} &= \frac{1-x^{k-1}}{v_2}; \\ \frac{x_L^* + \delta^k}{v_1} &= \frac{1-(x_L^* + \delta^{k-1})}{v_2}; \\ \delta^k &= -\frac{v_1}{v_2}\,\delta^{k-1}; \\ \delta^k &= (-1)^k\left(\frac{v_1}{v_2}\right)^k \delta^0. \end{aligned}$$

Since $v_1 < v_2$, as $k \rightarrow \infty$, $|\delta^k| \rightarrow 0$ and $x^k \rightarrow x_L^*$. □

Define the throughput of the system as the number of instances of the product produced per unit time.

Lemma 3.4. *If the subassembly lines are sequenced as $L \rightarrow R \rightarrow D$ in Region 1, then the throughput of the two-worker bucket brigade on the fixed point x_L^* is $v_1 + v_2$, the maximum possible.*

Proof. The throughput of the system on the fixed point x_L^* is

$$\begin{aligned} \tau_L &= \left(\frac{x_L^*}{v_1}\right)^{-1}; \\ &= v_1 + v_2. \end{aligned}$$

□

An alternate way in Region 1 is to sequence the subassembly lines as $R \rightarrow L \rightarrow D$ such as shown in Figure 3.3 (b).

Lemma 3.5. *If the subassembly lines are sequenced as $R \rightarrow L \rightarrow D$ in Region 1, then there exists a unique fixed point on line R and it is given by*

$$x_R^* = l + \frac{v_1}{v_1 + v_2}. \tag{3.2}$$

Proof. Following the order of lines $R \rightarrow L \rightarrow D$, the fixed point x_R^* is found by solving

$$\frac{x_R^* - l}{v_1} = \frac{1 - x_R^* + l}{v_2}.$$

Since $x_R^* = l + s$, $l < x_R^* < l + r$, the fixed point is located on line R. □

When the system operates on the fixed point x_R^* worker 1 hands an item off to worker 2 at x_R^* at each iteration. After relinquishing item k, worker 1 returns to node R to begin item $k+1$ while worker 2 continues to work on item k. When worker 2 finishes the work on line R for item k he proceeds to line L and then line D. Finally, when worker 2 completes item k at node E he returns to x_R^* to receive item $k+1$ from worker 1. Note that workers do not pass each other when they travel along the subassembly lines.

Lemma 3.6. *If the subassembly lines are sequenced as $R \rightarrow L \rightarrow D$ in Region 1, and $x^0 \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, then $x^k \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, for $k = 1, 2, 3, \ldots$.*

Proof. We first prove that if $x^{k-1} \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, then $x^k \in (l + max(0, 1 - (v_2/v_1)r), l + r]$. Following the order of lines $R \rightarrow L \rightarrow D$, we have

$$\begin{aligned} \frac{x^k - l}{v_1} &= \frac{1 - x^{k-1} + l}{v_2}; \\ x^k &= l + v_1 \left(\frac{1 - x^{k-1} + l}{v_2} \right). \end{aligned}$$

Case 1. If $1 - (v_2/v_1)r \geq 0$, $x^{k-1} \in (l + 1 - (v_2/v_1)r, l + r]$ implies

$$\begin{aligned} x^k &< l + v_1 \left[\frac{1 - [l + 1 - (v_2/v_1)r] + l}{v_2} \right]; \\ &= l + r; \end{aligned}$$

and

$$\begin{aligned} x^k &\geq l + v_1 \left[\frac{1 - (l + r) + l}{v_2} \right]; \\ &= l + v_1 \left[\frac{1 - r}{v_2} \right]; \\ &> l + 1 - \frac{v_2}{v_1} r. \end{aligned}$$

The last inequality is due to the fact that $s < r$. Thus, $x^k \in (l + 1 - (v_2/v_1)r, l + r]$.

Case 2. If $1 - (v_2/v_1)r < 0 \Rightarrow v_1/v_2 < r$, $x^{k-1} \in (l, l + r]$ implies

$$\begin{aligned} x^k &< l + v_1 \left(\frac{1 - l + l}{v_2} \right); \\ &< l + r; \end{aligned}$$

and

$$\begin{aligned} x^k &\geq l + v_1 \left[\frac{1 - (l + r) + l}{v_2} \right]; \\ &= l + v_1 \left[\frac{1 - r}{v_2} \right]; \\ &> l. \end{aligned}$$

Thus, $x^k \in (l, l + r]$. Since k is arbitrary we conclude that if $x^0 \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, then $x^k \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, for $k = 1, 2, 3, \ldots$. □

Lemma 3.6 shows that if $x^0 \in (l + max(0, 1 - (v_2/v_1)r), l + r]$ then the hand-off locations x^k are always on line R.

Lemma 3.7. *If the subassembly lines are sequenced as $R \rightarrow L \rightarrow D$ in Region 1, and $x^0 \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, then the two-worker bucket brigade converges to the fixed point x_R^*.*

Proof. According to Lemma 3.6, if $x^0 \in (l + max(0, 1 - (v_2/v_1)r), l + r]$ then the hand-off locations x^k are always on line R. Let $x^k = x_R^* + \delta^k$. Since

$$\begin{aligned} \frac{x^k - l}{v_1} &= \frac{1 - x^{k-1} + l}{v_2}; \\ \frac{x_R^* + \delta^k - l}{v_1} &= \frac{1 - (x_R^* + \delta^{k-1}) + l}{v_2}; \\ \delta^k &= -\frac{v_1}{v_2}\, \delta^{k-1}; \\ \delta^k &= (-1)^k \left(\frac{v_1}{v_2}\right)^k \delta^0. \end{aligned}$$

Since $v_1 < v_2$, as $k \rightarrow \infty$, $|\delta^k| \rightarrow 0$ and $x^k \rightarrow x_R^*$. □

Lemma 3.8. *If the subassembly lines are sequenced as $R \rightarrow L \rightarrow D$ in Region 1, then the throughput of the two-worker bucket brigade on the fixed point x_R^* is $v_1 + v_2$, the maximum possible.*

Proof. The throughput of the system on the fixed point x_R^* is

$$\begin{aligned} \tau_R &= \left(\frac{x_R^* - l}{v_1}\right)^{-1}; \\ &= v_1 + v_2. \end{aligned}$$

□

3.5.2 *Region 2* $(l < s < r)$

As s becomes larger than l the fixed point x_L^* (Eq. (3.1)) on line L no longer exists. There is only one way to order the subassembly lines to achieve balance: $R \rightarrow L \rightarrow D$. Figure 3.3 (b) shows the ordering of subassembly lines.

Lemma 3.9. *If the subassembly lines are sequenced as $R \to L \to D$ in Region 2, then there exists a unique fixed point on line R and it is given by*

$$x_R^* = l + \frac{v_1}{v_1 + v_2}. \tag{3.3}$$

Proof. The proof is the same as that of Lemma 3.5. □

Upon the fixed point x_R^* each worker executes a fixed portion of the total work content of an item. After a hand-off worker 1 returns to node R to begin a new item while worker 2 continues the work received from worker 1. When worker 2 finishes the work on line R he proceeds to line L and then line D. Finally, when worker 2 completes an item at node E he returns to x_R^* to receive new work from worker 1.

Lemma 3.10. *If the subassembly lines are sequenced as $R \to L \to D$ in Region 2, and $x^0 \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, then $x^k \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, for $k = 1, 2, 3, \ldots$.*

Proof. The proof is the same as that of Lemma 3.6. □

Lemma 3.10 ensures that if $x^0 \in (l+max(0, 1-(v_2/v_1)r), l+r]$, then the hand-off locations x^k are always on line R.

Lemma 3.11. *If the subassembly lines are sequenced as $R \to L \to D$ in Region 2, and $x^0 \in (l + max(0, 1 - (v_2/v_1)r), l + r]$, then the two-worker bucket brigade converges to the fixed point x_R^*.*

Proof. The proof is the same as that of Lemma 3.7. □

Lemma 3.12. *If the subassembly lines are sequenced as $R \to L \to D$ in Region 2, then the throughput of the two-worker bucket brigade on the fixed point x_R^* is $v_1 + v_2$, the maximum possible.*

Proof. The proof is the same as that of Lemma 3.8. □

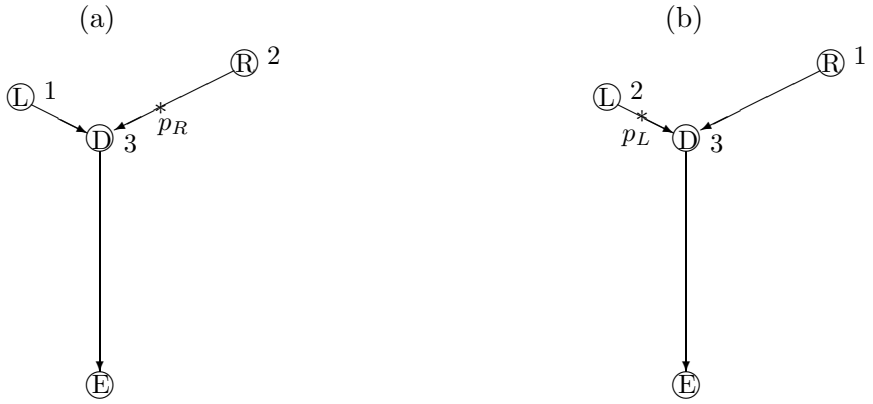

Figure 3.4: (a) For an item begun from node L, subassembly lines are sequenced as $L \to R \to D$. The item is handed off at p_R. (b) For an item begun from node R, subassembly lines are sequenced as $R \to L \to D$. The item is handed off at p_L.

3.5.3 *Region 3* $(r < s < l + r)$

In this region there exists no ordering of subassembly lines in which the two-worker bucket brigade converges to a fixed point. Thus, the simple assembly in-tree cannot be serialized to achieve balance. However, a period-2 orbit or 2-cycle [2, 19] is found if the following sequencing scheme is adopted. Imagine that, when a new item is begun, it is assigned an order of subassembly lines and this order is revealed to each worker when they take over production of the item. Furthermore, if we assign the order $L \to R \to D$ ($R \to L \to D$) to an item begun from node L (R), then there is a period-2 orbit on the in-tree.

When the system operates on the period-2 orbit the hand-offs occur periodically at two fixed locations p_L and p_R, where p_L and p_R are on lines L and R respectively. Figure 3.4 shows these hand-off locations. Suppose item k is begun from node L such as shown in Figure 3.4 (a). Following the order $L \to R \to D$, worker 1 first completes the work on line L and continues on line R for item k. The kth hand-off occurs when worker 2 receives item k from worker 1 at p_R. After relinquishing item k worker 1 walks back to node R to begin item $k + 1$ while worker 2 continues to work on item k. When worker 2 finishes the work on line R for item k, he proceeds to line D. Meanwhile, worker 1 is working on item $k + 1$ on line R. Following the new order $R \to L \to D$ shown in Figure 3.4 (b), worker 1

first finishes the work on line R and then proceeds to line L to work for item $k+1$ before worker 2 completes item k.

The $(k+1)$st hand-off occurs when worker 2 receives item $k+1$ from worker 1 at p_L. After the hand-off worker 1 walks back to node L to initiate item $k+2$ while worker 2 continues to work on item $k+1$. Following the order $L \to R \to D$ again worker 1 will finish the work on line L and proceed to line R for item $k+2$ before worker 2 completes item $k+1$. The $(k+2)$nd hand-off will again occur at p_R. Consequently, the hand-offs of the two-worker system repeatedly occur at two alternate locations p_L and p_R. Note that workers do not pass each other when they travel along the subassembly lines.

Lemma 3.13. *If the order of subassembly lines $L \to R \to D$ ($R \to L \to D$) is assigned to an item begun from node L (R) in Region 3, then there exists a unique period-2 orbit p_L and p_R, where p_L and p_R are on lines L and R respectively, and they are given by*

$$p_L = \frac{v_1}{v_1+v_2} - r; \tag{3.4}$$

$$p_R = \frac{v_1}{v_1+v_2}. \tag{3.5}$$

Proof. Since the hand-offs occur periodically at p_L and p_R such as shown in Figure 3.4, following the ordering of subassembly lines for each item, p_L and p_R can be found by solving

$$\frac{p_R}{v_1} = \frac{1-p_L-r}{v_2};$$

and

$$\frac{r+p_L}{v_1} = \frac{1-p_R}{v_2}.$$

Furthermore, $p_L = s-r$, $p_R = s$, and $r < s < l+r$ imply that $0 < p_L < l$ and $r < p_R < l+r$. Thus, p_L is on line L and p_R is on line R. □

Lemma 3.14. *If the order of subassembly lines $L \to R \to D$ ($R \to L \to D$) is assigned to an item begun from node L (R) in Region 3, then the throughput of the two-worker bucket brigade on the period-2 orbit p_L and p_R is v_1+v_2, the maximum possible.*

Proof. The throughput of the system on the period-2 orbit p_L and p_R is

$$\begin{aligned} \tau_2 &= \frac{2}{\frac{p_R}{v_1} + \frac{r+p_L}{v_1}}; \\ &= v_1 + v_2. \end{aligned}$$

□

3.5.4 *Region 4* $(s > l + r)$

In this region there are two ways to order the subassembly lines: $L \to R \to D$ and $R \to L \to D$. Both orders result in a fixed point on line D.

Lemma 3.15. *If the subassembly lines are sequenced as $L \to R \to D$ or $R \to L \to D$ in Region 4, then there exists a unique fixed point on line D and it is given by*

$$x_D^* = \frac{v_1}{v_1 + v_2}. \tag{3.6}$$

Proof. Following the order of lines $L \to R \to D$ or $R \to L \to D$, the fixed point x_D^* is found by solving

$$\frac{x_D^*}{v_1} = \frac{1 - x_D^*}{v_2}.$$

Since $x_D^* = s > l + r$, the fixed point is located on line D. □

When the system operates on the fixed point x_D^* worker 1 hands an item off to worker 2 at x_D^* at each iteration. After relinquishing item k worker 1 returns to node L (R), if order $L \to R \to D$ $(R \to L \to D)$ is used, to begin item $k+1$ while worker 2 continues to work on item k. When worker 1 finishes the work on line L (R) for item $k+1$ he proceeds to line R (L) and then line D. The next hand-off occurs when worker 2 finishes the work on line D for item k and returns to x_D^* to receive item $k+1$ from worker 1. Note that workers do not pass each other when they travel along the subassembly lines.

Lemma 3.16. *If the subassembly lines are sequenced as $L \to R \to D$ or $R \to L \to D$ in Region 4, and $x^0 \in (l + r, 1 - (v_2/v_1)(l + r))$, then $x^k \in (l + r, 1 - (v_2/v_1)(l + r))$, for $k = 1, 2, 3, \ldots$.*

Proof. We first prove that if $x^{k-1} \in (l+r, 1-(v_2/v_1)(l+r))$, then $x^k \in (l+r, 1-(v_2/v_1)(l+r))$. Following the order of lines $L \to R \to D$ or $R \to L \to D$, we have

$$\begin{aligned} \frac{x^k}{v_1} &= \frac{1-x^{k-1}}{v_2}; \\ x^k &= v_1\left(\frac{1-x^{k-1}}{v_2}\right). \end{aligned}$$

$x^{k-1} \in (l+r, 1-(v_2/v_1)(l+r))$ implies

$$\begin{aligned} x^k &> v_1\left[\frac{1-[1-(v_2/v_1)(l+r)]}{v_2}\right]; \\ &= l+r; \end{aligned}$$

and

$$\begin{aligned} x^k &< v_1\left[\frac{1-(l+r)}{v_2}\right]; \\ &< 1-\frac{v_2}{v_1}\,(l+r). \end{aligned}$$

The last inequality is due to $s > l+r$. Thus, $x^k \in (l+r, 1-(v_2/v_1)(l+r))$. Since k is arbitrary we conclude that if $x^0 \in (l+r, 1-(v_2/v_1)(l+r))$, then $x^k \in (l+r, 1-(v_2/v_1)(l+r))$, for $k = 1, 2, 3, \ldots$. □

Lemma 3.16 shows that if $x^0 \in (l+r, 1-(v_2/v_1)(l+r))$ then the hand-off locations x^k are always on line D.

Lemma 3.17. *If the subassembly lines are sequenced as $L \to R \to D$ or $R \to L \to D$ in Region 4, and $x^0 \in (l+r, 1-(v_2/v_1)(l+r))$, then the two-worker bucket brigade converges to the fixed point x_D^*.*

Proof. According to Lemma 3.16, if $x^0 \in (l+r, 1-(v_2/v_1)(l+r))$ then the hand-off

locations x^k are always on line D. Let $x^k = x_D^* + \delta^k$. Since

$$\begin{aligned}
\frac{x^k}{v_1} &= \frac{1 - x^{k-1}}{v_2}; \\
\frac{x_D^* + \delta^k}{v_1} &= \frac{1 - (x_D^* + \delta^{k-1})}{v_2}; \\
\delta^k &= -\frac{v_1}{v_2}\, \delta^{k-1}; \\
\delta^k &= (-1)^k \left(\frac{v_1}{v_2}\right)^k \delta^0.
\end{aligned}$$

Since $v_1 < v_2$, as $k \to \infty$, $|\delta^k| \to 0$ and $x^k \to x_D^*$. □

Lemma 3.18. *If the subassembly lines are sequenced as $L \to R \to D$ or $R \to L \to D$ in Region 4, then the throughput of the two-worker bucket brigade on the fixed point x_D^* is $v_1 + v_2$, the maximum possible.*

Proof. The throughput of the system on the fixed point x_D^* is

$$\begin{aligned}
\tau_D &= \left(\frac{x_D^*}{v_1}\right)^{-1}; \\
&= v_1 + v_2.
\end{aligned}$$

□

Thus, as s increases (as the work velocity of worker 1 approaches that of worker 2) with respect to l and r, and if the subassembly lines are serialized into a correct order, then the stationary state of the two-worker bucket brigade changes from two fixed points x_L^* (Eq. (3.1)) and x_R^* (Eq. (3.2)) on lines L and R respectively, to a single fixed point x_R^* (Eq. (3.3)) on line R, a period-2 orbit p_L (Eq. (3.4)) and p_R (Eq. (3.5)), and a fixed point x_D^* (Eq. (3.6)) on line D. Furthermore, the throughput of the system on every stationary state is equal to $v_1 + v_2$.

3.6 Summary

Any fixed point is preferable to the period-2 orbit for the two-worker bucket brigade on the simple assembly in-tree because the system exhibits simple dynamic behavior on a

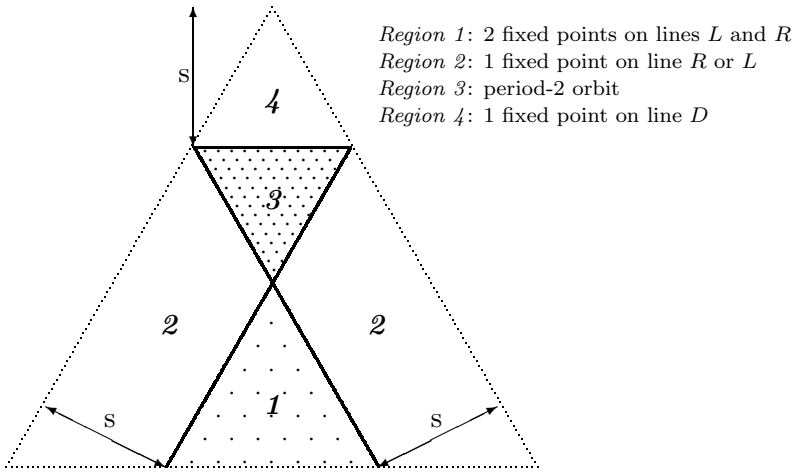

Figure 3.5: Given the value of s (workers' work velocities), the stationary states of the two-worker bucket brigade are characterized by different areas for the location of node D within the triangle in Figure 3.2.

fixed point. That is, each worker repeatedly executes the same portion of work content of each instance of the product produced. Thus, the skills of each worker is reinforced by repetition. Furthermore, handing the items off at a fixed location also causes less confusions and errors in the operation.

Given the work velocities of workers (the value of s), what is the layout for the assembly in-tree so that the two-worker bucket brigade converges to a fixed point? When s is given, different regions described in Section 3.5 correspond to different values of l and r, which determine the location of node D in Figure 3.2. Each region described in Section 3.5 corresponds to a distinct area for the location of node D within the left portion of the triangle in Figure 3.2. Different areas, each of which is characterized by some stationary state(s), are illustrated in Figure 3.5. Due to the symmetry of the 'Y' shape in-tree, similar dynamics is found in layouts with $l > r$. Thus, Figure 3.5 has symmetrical results on the right portion. Region 1 has two fixed points, one on line L and the other on line R. Region 2 corresponds to two separate areas in Figure 3.5. The area on the left (where $l \leq r$) in Figure 3.5 has a fixed point on line R; and the area on the right (where $l > r$) has a fixed

point on line L. Region 3 has a period-2 orbit on which the hand-offs occur on lines L and R periodically. Finally, Region 4 has a fixed point on line D. Note that the ratio of the area of Region 3 to that of the entire triangle is s^2. This ratio increases with s but is bounded above by $1/4$ because $s < 1/2$.

The idea developed here can be generalized to any restrictive in-trees with m subassembly lines and a team of n workers. Given an in-tree and a team of workers, we first sequence the workers from slowest to fastest and serialize the work content on the in-tree. Upon constructing a serial order of subassembly lines we check whether each worker can walk back, along the subassembly lines, to the corresponding hand-off location in Equation 1.1. If the serial order of subassembly lines does not allow some worker to walk back to get more work from his predecessor, then we try another serial order. We repeat this procedure until a feasible serial order of subassembly lines (if any) is found.

Chapter 4

Stability and Chaos on A Bucket Brigade Assembly Line

A model of bucket brigade assembly lines, in which workers spend significant time to walk back for more work and they are allowed to pass each other, may exhibit chaotic behavior. Even in the case with only two workers, the dynamic behavior of this purely deterministic system can be effectively random if the system is improperly configured. Spatially, the chaotic behavior is characterized by erratic hand-off locations that may spread over almost the entire assembly line. Temporally, the chaotic behavior is reflected by large variability in completion times of items. We show analytically how the dynamics of the system undergoes a transition from stability to chaotic behavior as the velocities of workers change.

4.1 Introduction

The existence of variability often degrades the performance of manufacturing systems (see, for example, Chapter 9 of [22]). For instance, the variation in processing times of jobs at a work station on an assembly line may cause the work-in-process and the flow time of the line to increase and thus, reduces service level. As a common practice, factory managers strive to eradicate variability from their shop floors. To achieve this goal and to have

effective manufacturing management, it is crucial to understand and manage variability in a manufacturing system.

The causes of variability in manufacturing systems can be attributed to *random variation*, which is a result of events beyond our immediate control [22]. For instance, the processing times of jobs at a work station may fluctuate with the motivation of a worker. Similarly, the inter-arrival times between customer orders are, in general, unpredictable. On the other hand, variability can also be due to *controllable variation*, which occurs as a direct result of human decisions [22]. For instance, if several products are produced by an assembly line, there will be variability in their physical dimensions and processing time. Similarly, the content of a buffer in front of a work station changes more drastically if items are moved in batches than if moved one at a time.

Variability observed in a manufacturing system is often a consequence of one type of variation superimposed on the other. Identifying the underlying causes of variability with respect to these two types of variation can be challenging. Since random variation is beyond our control, in general, it is difficult to eliminate. Controllable variation, on the other hand, can be reduced by making proper decisions on the system.

It has been found that many simple, deterministic systems may generate surprisingly irregular, complicated dynamic behavior. These systems are often called *chaotic dynamical systems* or simply *chaotic systems* [2, 19]. This suggests that we may sometimes understand irregular and complicated phenomena, such as that observed on the shop floors, in terms of simple models. In the context of manufacturing systems, the controllable variation observed on a system may sometimes be a manifestation of the chaotic dynamics of the system under some improper control policies. As a result, analyzing the dynamics of a model under certain control policies may help us understand the impact of these policies on the resultant variability.

Here, we consider a generalized model of bucket brigades [10], which is a deterministic model of workers sharing work on assembly lines. Focusing our study on a deterministic

model allows us to decouple the effects of controllable variation from that of random variation. The model includes no stochastic elements and thus allows us to see how the behavior of the system evolves without any "noise". Using concepts and techniques from the theory of dynamical systems [2, 19], we show that the dynamics of this purely deterministic model can be chaotic and may induce large variability if the system is improperly configured. Our results demonstrate that variability observed on a manufacturing system may be reduced or managed through studying the dynamics of a simple, deterministic model.

4.2 Literature review

The most well-known model in manufacturing systems that exhibits chaotic dynamics is called the switched arrival system [17]. It is a fluid model of a manufacturing system in which a single switching server distributes work over N parallel machines. The amount of work in the buffer in front of each machine is assumed to be a continuous variable. The processing rate at each machine is assumed to be constant. The server continues to fill a buffer until another buffer empties. The rate at which the server fills a buffer is equal to the sum of the processing rates of all machines. When the system is sampled at the times when a buffer empties, the dynamics of the system can be represented by a Poincaré map [2] which maps the interval $[0, 1]$ into itself. Chase *et al.* [17] showed that the switched arrival system can be chaotic and this behavior is reflected by the irregular fluctuation in the work amount in each buffer. More detailed models of the switched arrival system have been studied. These models consider finite buffer sizes, setup times for the server to switch from one buffer to another, discrete work content, etc. (see [4, 28] and references therein).

We will show in the following sections that a generalized model of bucket brigade assembly may exhibit chaotic behavior under a certain condition. Unlike the switched arrival system, which represents an idealized manufacturing system, bucket brigades are widely used in industry as a way of sharing work among workers on assembly lines such as the assembly of sewn products [10] and order-picking in warehouses [9, 12]. We shall

point out that both Chase *et al.* [17] and us base results on the work of Li and Yorke [25], who analyzed chaotic behavior in piecewise continuous functions with a finite number of discontinuities.

Bartholdi and Eisenstein [11] introduced a generalized model in which workers can pass each other (thus, they are not restricted to a fixed sequence along the line). Furthermore, each worker walks back with a finite velocity to receive work. This generalized model is more accurate in some settings where workers are allowed to pass each other and the time to walk back to get more work is not significantly smaller than the time to assemble the product. They found a new, generalized condition for a bucket brigade to converge to a fixed point. Here, we show that when this condition is violated the system may behave chaotically. We study the two-worker system and analyze its dynamics with respect to the velocities of workers.

4.3 A generalized model of bucket brigades

As in the Normative Model, we make the following assumption on the work content of the product assembled by a bucket brigade assembly line.

Assumption 4.1. (Smoothness and Predictability of Work)
The work content of the product is a constant (which we normalize to 1); and it is distributed continuously and uniformly over the assembly line. Furthermore, the work content at any work station is preëmptible without significant loss of work.

An example of such an assembly line is one with relatively many work stations, each of which has a small piece of the total work content. As discussed in [8, 10, 12] this assumption is not strictly necessary for a bucket brigade to be effective in practice but it greatly simplifies the analysis. We will show that even with this simplification the dynamical behavior of the line can be very complicated if it is improperly configured.

Furthermore, suppose we have a bucket brigade with n workers and they are indexed

Table 4.1: The Bucket Brigade Rules determine what each individual team member should do. Under these rules workers are not restricted to a fixed sequence along the line.

Forward Rule: Work forward with your item until 1. your item is handed off to a co-worker; or 2. you complete your item; then follow the Backward Rule.
Backward Rule: Walk back to get more work, 1. if you encounter a worker $j < i$ working forward then take over his item; 2. otherwise, begin a new item at the start of the line; then follow the Forward Rule.

from 1 to n. We make the following assumption on the velocities of workers.

Assumption 4.2. (Workers Are Characterized by Forward and Backward Velocities)

Each worker i is characterized by a constant forward (work) velocity v_i, and a constant backward (walk-back) velocity w_i. Both v_i and w_i are positive, for all $i = 1, \ldots, n$.

In contrast, in the Normative Model of bucket brigades [8, 10] each worker walks back with an infinite velocity. In some settings the assumption of infinite walk-back velocities may be inappropriate. For example, in low-density order-picking workers walk a considerable distance between picks. Thus, the time to *work forward* to perform a few picks is not much greater than the time to *walk back* to get more work.

Each worker i in the generalized model follows two simple rules stated in Table 4.1. Under these rules workers are not restricted to a fixed sequence along the line.

To facilitate discussion and avoid ambiguity we say a forward (backward) *overtaking* occurs when one worker overtakes another worker while both of them are working forward (walking back). On the other hand, we say a *passing* occurs when two workers pass each other while they are moving in opposite directions. Both overtaking and passing are

possible under the rules stated in Table 4.1. Furthermore, a *hand-off* occurs when one worker takes over an item from another worker.

For this generalized model it is shown in [11] that the bucket brigade converges to a unique fixed point if workers are indexed such that the following condition holds.

Stability Condition:

$$\frac{1}{v_{i-1}} - \frac{1}{w_{i-1}} \quad > \quad \frac{1}{v_i} - \frac{1}{w_i}. \tag{4.1}$$

At the fixed point workers are sequenced along the line according to their indices and no overtaking or passing occurs. The behavior on the fixed point is similar to that of the Normative Model [8, 10]: every worker repeats a segment of work content on each item produced (see top left of Figure 4.1 for an example with two workers). Each successive item is produced after a constant time interval (see top right of Figure 4.1). This simplifies the downstream processes.

If the Stability Condition (4.1) does not hold then simulation results on two-worker lines suggest that the hand-offs between workers do not settle down at a fixed location. Instead, the hand-off location changes from one point to another on the assembly line in an erratic manner. More specifically, the hand-off locations distribute over several subintervals along the assembly line (see bottom left of Figure 4.1). Each of the subintervals is "visited" by the hand-offs with a certain frequency. Furthermore, the completion times of items become irregular (see bottom right of Figure 4.1). This demonstrates that large variability can be induced by the dynamics of a purely deterministic system.

Our main contribution here is to prove that if the Stability Condition (4.1) is violated then the two-worker system behaves chaotically according to the definition in [2]. We also demonstrate that the chaotic behavior exhibited by this deterministic system is effectively indistinguishable from randomness.

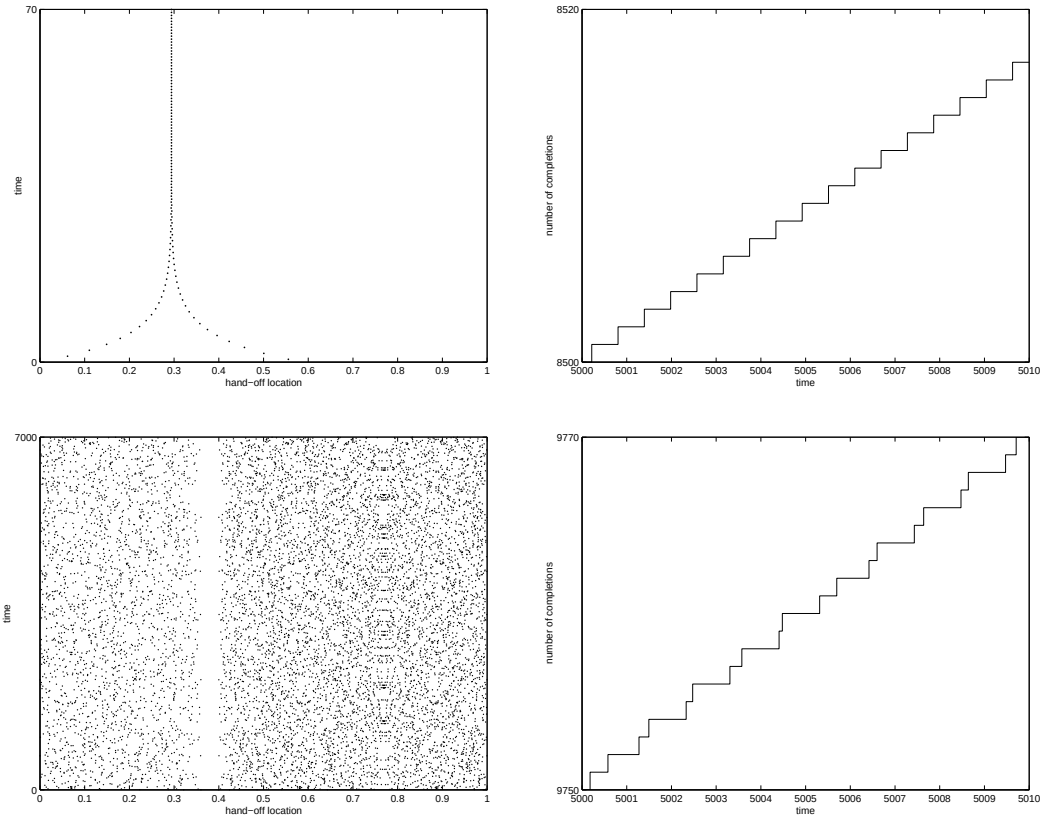

Figure 4.1: **Stability and chaos:** The top graphs show the stable behavior of a two-worker line when the Stability Condition (4.1) holds. Hand-off locations converge quickly to a single point (top left) and each successive item is produced in a fixed time interval (top right). The bottom graphs show the chaotic behavior of the line when the Stability Condition is violated. Hand-off locations spread erratically along the assembly line (bottom left) and completion times become irregular (bottom right). This demonstrates that large variability can be induced by the dynamics of a purely deterministic system. In all graphs $w_1 = 1, v_2 = 3, w_2 = 2$, and $v_1 = 1$ and 3 in top and bottom graphs respectively.

4.4 Invariant measures

Now we present the groundwork for analysis of chaotic behavior. The dynamics of the two-worker bucket brigade can be described by a Poincaré map (or simply map) $x^{k+1} = f(x^k)$, where x^k represents the location of the kth hand-off on the assembly line. Since $x^k \in [0,1]$, the function f maps the closed interval $I = [0,1]$ into itself. It determines the subsequent iterates $x^k, k = 1, 2, 3, \ldots$, given the initial hand-off location x^0. The set $\{x^0, x^1, x^2, \ldots\}$ is called the *orbit* of x^0 under f [2].

Following standard terminology [2] we adopt the following definitions. We call x^* a fixed point if $x^* = f(x^*)$. We call x' a period-k point (or periodic point with period k) if $x' = f^k(x')$ and if k is the smallest such positive integer. The orbit with initial point x', which consists of k distinct points, is called a period-k orbit (or periodic orbit with period k). An orbit $\{x^0, x^1, x^2, \ldots, x^t, \ldots\}$ is called *asymptotically periodic* if it converges to a periodic orbit as $t \to \infty$.

Definition 4.1. *An orbit $\{x^0, x^1, x^2, \ldots\}$ is chaotic [2] if*

- *it is neither periodic nor asymptotically periodic; and*
- *the Lyapunov number $L(x^0) \equiv \lim_{t\to\infty}(f'(x^0)\cdots f'(x^{t-1}))^{1/t}$ exists and is greater than 1.*

Definition 4.2. *The forward limit set [2] of an orbit $\{x^0, x^1, x^2, \ldots\}$ is defined as*

$$\Omega(x^0) = \left\{x : \textit{for all } T \textit{ and } \epsilon \textit{ there exists } t > T \textit{ such that } |f^t(x^0) - x| < \epsilon\right\}.$$

The forward limit set of an orbit is the set of points which the orbit approaches arbitrarily closely, infinitely often. An *attractor* is a forward limit set which attracts a set of initial points that has nonzero length. Suppose the orbit $\{x^0, x^1, x^2, \ldots\}$ is chaotic. If $x^0 \in \Omega(x^0)$ then $\Omega(x^0)$ is called a *chaotic set* [2].

Definition 4.3. *A chaotic attractor [2] is a chaotic set that is also an attractor.*

Fixed points and periodic points contain a finite number of points. They can be considered as individual points when we study them. However, a chaotic attractor contains infinitely many points. We cannot keep track of individual points when we study a chaotic attractor. Instead, we describe a chaotic attractor by specifying how much of the attractor is in an interval. Consider an interval $S \subseteq I$. Define $\mu(S)$ as the *measure* of S. Intuitively, the measure of an interval is a number that represents "how much" a chaotic attractor is in the interval (see [2, 24] for an introduction). The measure of an interval has the following properties [2]:

- The measure of any interval is a nonnegative number.
- The measure of a disjoint union of a finite or countably infinite number of intervals is equal to the sum of the measures of the individual intervals.
- The measure of $I = [0, 1]$ is equal to 1.

Furthermore, μ is called an *invariant measure* with respect to f if $\mu(S) = \mu(f^{-1}(S))$, for every closed interval S. If the invariant measure μ of a closed interval S is given by

$$\mu(S) \quad = \quad \int_S p(x)\ dx,$$

then p is called the *density* of μ.

We observe that when the Stability Condition (4.1) is violated the orbit of almost every $x \in [0, 1]$ under f converges to a chaotic attractor (except for the marginal case where $1/v_1 - 1/w_1 = 1/v_2 - 1/w_2$ in which the orbits converge to periodic points). The asymptotic distribution of iterates of f over the interval I can be represented by a density function p. In other words, the hand-offs occur in a chaotic manner along the assembly line.

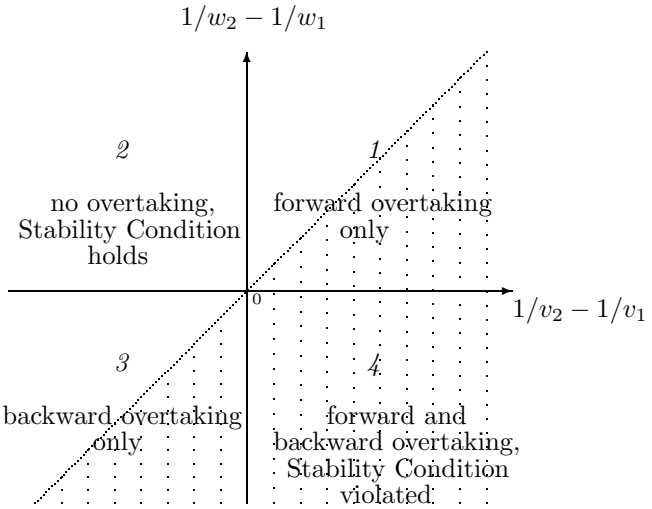

Figure 4.2: The dynamic behavior of the two-worker system after a transient period is described in four disjoint regions. The Stability Condition (4.1) is violated in the shaded area below the diagonal line where the system behaves chaotically.

4.5 Dynamics of two-worker lines

We describe the dynamics after a transient period as well as the asymptotic behavior of the two-worker system. Workers are indexed as 1 and 2. Worker i has forward (backward) velocity v_i (w_i), $i = 1, 2$. Suppose workers start from arbitrary locations.

Lemma 4.1. *In the two-worker system, worker 2 forward overtaking worker 1 is transient.*

Proof. Worker 2 must complete his initial item and thus the system will quickly be in a state where worker 2 is downstream of worker 1. Then, for worker 2 to ever forward overtake worker 1 the former must be upstream; so worker 2 must backward overtake worker 1 as both are walking back, and must therefore begin a new item before worker 1 reaches the start of the line. Then, worker 2 working forward must pass worker 1 walking back, and the system is again in a state where worker 2 is downstream of worker 1. □

Corollary 4.1. *In the two-worker system, after a transient period, worker 1 will complete his item after each forward overtaking.*

Proof. According to Lemma 4.1, after a transient period, any forward overtaking occurs only when worker 1 forward overtakes worker 2. Every time worker 1 forward overtakes worker 2 the former will complete his item. □

By similar arguments we have the following results.

Lemma 4.2. *In the two-worker system, worker 1 backward overtaking worker 2 is transient.*

Corollary 4.2. *In the two-worker system, after a transient period, worker 2 will begin a new item and will complete the item after each backward overtaking.*

We describe the dynamics of the two-worker system after a transient period in each of the following four mutually exclusive regions.

Region 1 $(v_1 > v_2; w_1 \geq w_2)$**:**

After a transient period, only forward overtaking can occur.

Region 2 $(v_1 \leq v_2; w_1 \geq w_2)$**:**

After a transient period, the workers will remain in sequence because neither forward nor backward overtaking can occur. Note that the Stability Condition (4.1) always holds in this region except for the marginal case where $v_1 = v_2$ and $w_1 = w_2$.

Region 3 $(v_1 \leq v_2; w_1 < w_2)$**:**

After a transient period, only backward overtaking can occur.

Region 4 $(v_1 > v_2; w_1 < w_2)$**:**

Both forward and backward overtaking can occur in this region even after a transient period. Note that the Stability Condition (4.1) is always violated in this region.

Furthermore, after a transient period, passing is possible in all regions except for Region 2. The dynamic behavior of the two-worker system after this transient period is summarized in Figure 4.2. Due to the fact that the system has different dynamic behavior in different

regions, we derive the map for each region separately. Recall that x^k is the kth hand-off location on the assembly line. $x^k \in [0,1]$ for $k = 0, 1, 2, \ldots$. Let $\lfloor x \rfloor$ represent the largest integer smaller or equal to x. Define $\tau_1 \equiv 1/v_1 + 1/w_1, \tau_2 \equiv 1/v_2 + 1/w_2, \alpha \equiv 1/v_1 + 1/w_2$, and $\beta \equiv 1/v_2 + 1/w_1$. Note that τ_1 (τ_2) represents the time for worker 1 (2) to walk back from the end of the line, begin a new item, and complete the item.

4.5.1 *Region 1* $(v_1 > v_2; w_1 \geq w_2)$

In this region only forward overtaking can occur after a transient period. In Figure 4.2 the Stability Condition (4.1) holds only in the northwest half of this quadrant.

Lemma 4.3. *The map $x^{k+1} = f(x^k)$ for the two-worker system in Region 1 is given by*

$$f(x^k) = \begin{cases} \frac{\tau_2}{\alpha} - \frac{(\overline{r}+1)\tau_1}{\alpha} - \frac{\beta}{\alpha}\, x^k, & \text{if } x^k \in [0, P_{\overline{r}}); \\ \frac{\tau_2}{\alpha} - \frac{r\tau_1}{\alpha} - \frac{\beta}{\alpha}\, x^k, & \text{if } x^k \in [P_r, P_{r-1}), \\ & \quad r = \overline{r}, \overline{r}-1, \ldots, 1; \\ \frac{\tau_2}{\alpha} - \frac{\beta}{\alpha}\, x^k, & \text{if } x^k \in [P_0, 1]; \end{cases} \tag{4.2}$$

where

$$P_r \;=\; \frac{1/v_2 - 1/v_1 - r(1/v_1 + 1/w_1)}{1/v_2 + 1/w_1}$$

and

$$\overline{r} \;=\; \left\lfloor \frac{1/v_2 - 1/v_1}{1/v_1 + 1/w_1} \right\rfloor.$$

Proof. See Appendix A.1. □

The map in Region 1 is piecewise linear with $\overline{r}+1$ discontinuities. Figure 4.3 shows the map in Region 1 with two different values of w_2, while other velocities are fixed. The top graph shows the map when the Stability Condition (4.1) holds. The dashed line represents the diagonal line $x^{k+1} = x^k$. The intersection of the diagonal with the function f gives the fixed point of f. Note that there exists a unique fixed point $x_0^* \approx 0.7$ that is *attracting* because $|f'(x_0^*)| = \beta/\alpha < 1$ [2]. The last inequality is due to the Stability Condition (4.1).

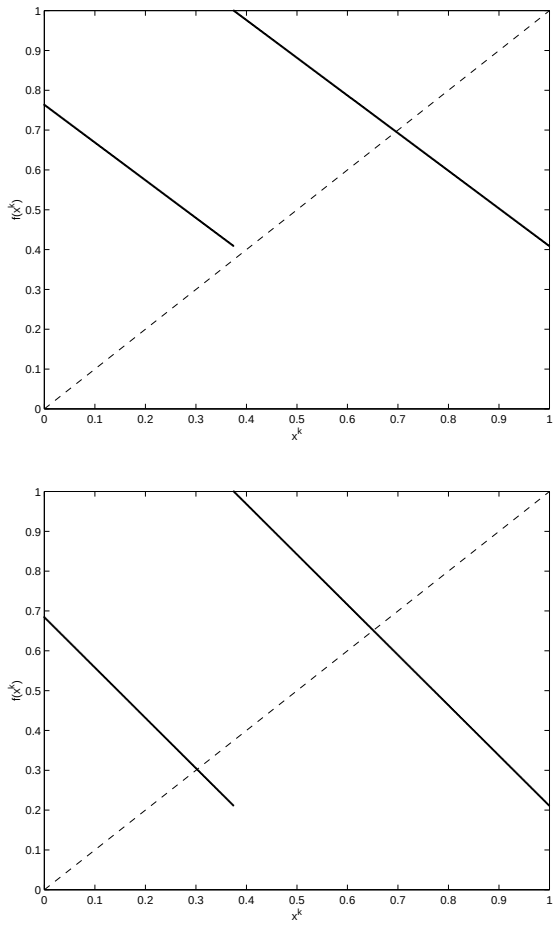

Figure 4.3: The map $f(x^k)$ in Region 1 is plotted with different values of w_2. In both graphs $v_1 = 2, w_1 = 3$, and $v_2 = 1$. The top graph shows the map when $w_2 = 1.1$ and so the Stability Condition (4.1) holds; and the bottom graph shows the map when $w_2 = 1.8$ and so the Stability Condition is violated.

The bottom graph shows the case when the Stability Condition is violated. In addition to $x_0^* \approx 0.7$, there is another fixed point $x_1^* \approx 0.3$. Both fixed points are *repelling* because $|f'(x_0^*)| = |f'(x_1^*)| = \beta/\alpha > 1$ [2]. Furthermore, we have the following observation.

Observation 4.1. *Except for the initial hand-off, all hand-offs in Region 1 occur in the interval* $[H_L, 1]$ *where*

$$H_L = \frac{1/w_2 - 1/w_1}{1/v_1 + 1/w_2}.$$

In Figure 4.3 the function f has only one discontinuity, at P_0 separating the horizontal axis into two intervals $[0, P_0)$ and $[P_0, 1]$. When the Stability Condition (4.1) holds $H_L > P_0$. Thus, in the top graph of Figure 4.3 the orbits under f are confined in the interval $[P_0, 1]$. Since x_0^* is attracting, $x^k \to x_0^*$ as $k \to \infty$. On the other hand, when the Stability Condition is violated $H_L \leq P_0$. Thus, in the bottom graph of Figure 4.3 the interval $[0, P_0)$ may be visited by the orbits. The orbits under f are repelled by x_0^* and x_1^*, and they travel in an erratic manner in the interval $[0, 1]$.

4.5.2 *Region 2* $(v_1 \leq v_2; w_1 \geq w_2)$

Neither forward nor backward overtaking can happen after a transient period in this region. This leads to the following lemma.

Lemma 4.4. *The map* $x^{k+1} = f(x^k)$ *for the two-worker system in Region 2 is given by*

$$f(x^k) = \frac{\tau_2}{\alpha} - \frac{\beta}{\alpha}\, x^k, \quad x^k \in [0, 1]. \tag{4.3}$$

Proof. See Appendix A.2. □

A fixed point x^* in this region is attracting because $|f'(x^*)| = \beta/\alpha < 1$ as the Stability Condition (4.1) always holds in this region (except for $v_1 = v_2$ and $w_1 = w_2$ in which case $|f'(x^*)| = 1$ and x^* becomes *neutral* [19]). Figure 4.4 shows an example of the map in this region. Since x^* is attracting, $x^k \to x^*$ as $k \to \infty$.

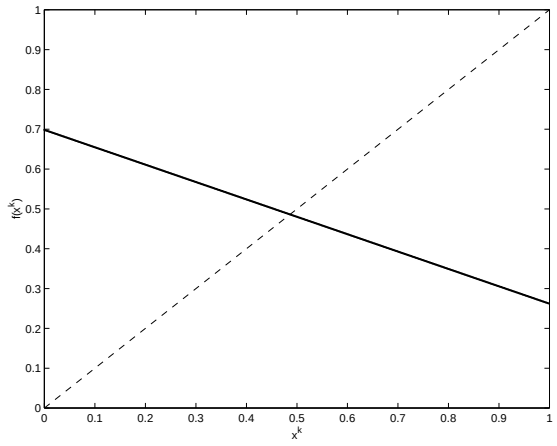

Figure 4.4: The map $f(x^k)$ in Region 2 with $v_1 = 1.1, w_1 = 2, v_2 = 3$, and $w_2 = 1$.

4.5.3 *Region 3* $(v_1 \leq v_2; w_1 < w_2)$

In this region only backward overtaking can occur after a transient period. In Figure 4.2 the Stability Condition (4.1) holds only in the northwest half of this quadrant.

Lemma 4.5. *The map $x^{k+1} = f(x^k)$ for the two-worker system in Region 3 is given by*

$$f(x^k) = \begin{cases} \frac{\tau_2}{\alpha} - \frac{\beta}{\alpha}\, x^k, & \text{if } x^k \in [0, P_0]; \\ \frac{(s+1)\tau_2}{\alpha} - \frac{\beta}{\alpha}\, x^k, & \text{if } x^k \in (P_{s-1}, P_s], \\ & \qquad s = 1, 2, \ldots, \overline{s}; \\ \frac{(\overline{s}+2)\tau_2}{\alpha} - \frac{\beta}{\alpha}\, x^k, & \text{if } x^k \in (P_{\overline{s}}, 1]; \end{cases} \tag{4.4}$$

where

$$P_s \;=\; \frac{(s+1)(1/v_2 + 1/w_2)}{1/v_2 + 1/w_1}$$

and

$$\overline{s} \;=\; \left\lfloor \frac{1/w_1 - 1/w_2}{1/v_2 + 1/w_2} \right\rfloor.$$

Proof. See Appendix A.3. □

Note that the indexing of P_s is the reverse of that of P_r in Region 1. This is because in Region 3 (as we show in the Appendix) if $x^k \in (P_{s-1}, P_s]$ then there will be s *backward*

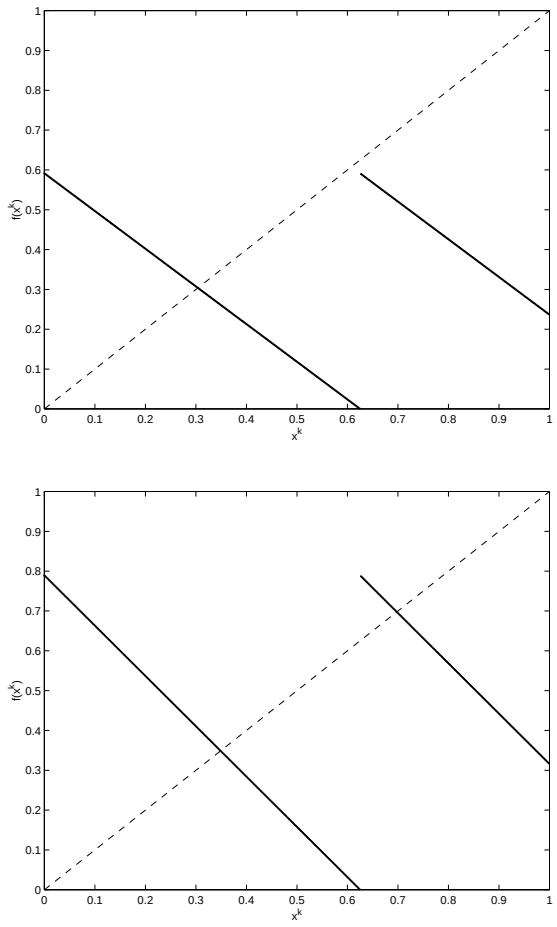

Figure 4.5: The map $f(x^k)$ in Region 3 is plotted with different values of v_1. In both graphs $w_1 = 1, v_2 = 3$, and $w_2 = 2$. The top graph shows the map when $v_1 = 1.1$ and so the Stability Condition (4.1) holds; and the bottom graph shows the map when $v_1 = 1.8$ and so the Stability Condition is violated.

overtakings before the $(k+1)$st hand-off. Thus, P_s increases as s increases. On the other hand, in Region 1 each subinterval $[P_r, P_{r-1})$ corresponds to r *forward* overtakings. Thus, P_r decreases as r increases in Region 1.

The map in Region 3 is piecewise linear with $\overline{s}+1$ discontinuities. Figure 4.5 shows the map with two different values of v_1, while other velocities are fixed. The top graph shows the map when the Stability Condition (4.1) holds. Note that there exists a unique fixed point $x_0^* \approx 0.3$ that is attracting because $|f'(x_0^*)| = \beta/\alpha < 1$. The bottom graph shows the case when the Stability Condition is violated. In addition to $x_0^* \approx 0.3$, there is another fixed point $x_1^* \approx 0.7$. Both fixed points are repelling because $|f'(x_0^*)| = |f'(x_1^*)| = \beta/\alpha > 1$. Furthermore, we have the following observation.

Observation 4.2. *Except for the initial hand-off, all hand-offs in Region 3 occur in the interval* $[0, H_U]$ *where*

$$H_U = \frac{1/v_2 + 1/w_2}{1/v_1 + 1/w_2}.$$

In Figure 4.5 the function f has only one discontinuity at P_0 separating the horizontal axis into two intervals $[0, P_0]$ and $(P_0, 1]$. When the Stability Condition (4.1) holds $H_U < P_0$. Thus, in the top graph of Figure 4.5 the orbits under f are confined in the interval $[0, P_0]$. Since x_0^* is attracting, $x^k \to x_0^*$ as $k \to \infty$. On the other hand, when the Stability Condition is violated $H_U \geq P_0$. Thus, in the bottom graph of Figure 4.5 the interval $(P_0, 1]$ may be visited by the orbits. The orbits under f are repelled by x_0^* and x_1^*, and they travel in an erratic manner in the interval $[0, 1]$.

4.5.4 *Region 4* $(v_1 > v_2; w_1 < w_2)$

The dynamics of two-worker bucket brigades in this region are more complicated than that of other regions because both forward and backward overtakings are possible in this region.

Let Z^+ be the set of nonnegative integers. Furthermore, let (a, b) denote an ordered pair, where $a, b \in Z^+$.

Definition 4.4. *For any nonnegative integers a, b, c, and d, we say (a, b) is **pairwise smaller** than (c, d) if and only if $a < c$ and $b < d$.*

Definition 4.5. *For a set S of ordered pairs of nonnegative integers, an ordered pair $(a, b) \in S$ is **pairwise minimal** in S if and only if (a, b) is pairwise smaller than (c, d), for all $(c, d) \in S \backslash \{(a, b)\}$.*

For any $x^k \in [0, 1]$, the location of the $(k+1)$st hand-off (if there is one) is determined by an (r^*, s^*) pair, as we show in the Appendix, which falls within the set

$$P(x^k) = \left\{ (r, s) : \frac{1}{v_2} - \beta x^k - \frac{1}{v_1} \leq r\tau_1 - s\tau_2 \leq \frac{1}{v_2} - \beta x^k + \frac{1}{w_2}, r, s \in Z^+ \right\}.$$

The map in this region is constructed in the following lemma.

Lemma 4.6. *Suppose $P(x^k)$ is nonempty for all $k = 0, 1, 2, \ldots$. The map $x^{k+1} = f(x^k)$ for the two-worker system in Region 4 is given by*

$$f(x^k) \quad = \quad \frac{(s^* + 1)\tau_2}{\alpha} - \frac{r^*\tau_1}{\alpha} - \frac{\beta}{\alpha}\, x^k, \quad x^k \in [0, 1]; \tag{4.5}$$

where (r^*, s^*) is pairwise minimal in the set $P(x^k)$.

Proof. See Appendix A.4. □

Note that in the above lemma we rule out the cases when $P(x^k)$ is empty for some $x^k \in [0, 1]$. These cases correspond to the situations in which there will be no hand-offs after the kth hand-off. In Equation (4.5) r^* and s^* depend on the value of x^k. Given x^k we determine x^{k+1} by finding (r^*, s^*) that is pairwise minimal in the set $P(x^k)$. This is equivalent to solving an integer program subject to the polyhedron that defines the set $P(x^k)$. As a result, the map $f(x^k)$ in this region cannot be expressed in closed form as a piecewise linear function of x^k as in other regions. However, the function $f(x^k)$ can be proved to be piecewise linear.

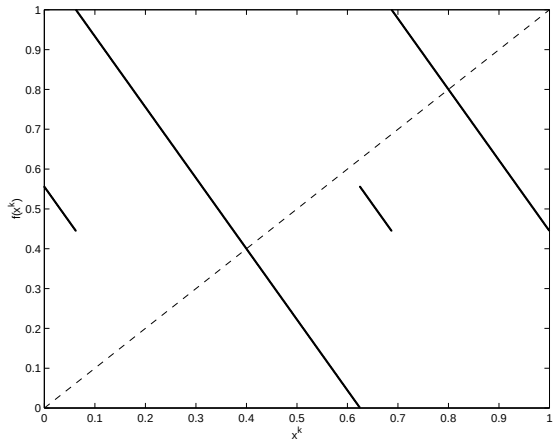

Figure 4.6: The map $f(x^k)$ in Region 4 with $v_1 = 4, w_1 = 1, v_2 = 3$, and $w_2 = 2$. There are two repelling fixed points.

Lemma 4.7. *Suppose $P(x^k)$ is nonempty for all $k = 0, 1, 2, \ldots$. The map $f(x^k)$ for the two-worker system in Region 4 is piecewise linear with a finite number of points of discontinuity of f and f'. Furthermore, except for the points of discontinuity, $f' = -\frac{\beta}{\alpha}$.*

Proof. See Appendix A.4. □

We plot the map numerically in Figure 4.6. There are two fixed points $x_0^* \approx 0.4$ and $x_1^* \approx 0.8$. Both are repelling because $|f'(x_0^*)| = |f'(x_1^*)| = \beta/\alpha > 1$ as the Stability Condition (4.1) is always violated in this region.

4.5.5 Asymptotic behavior

For the two-worker system we say the Stability Condition (4.1) is *strongly violated* if

$$\frac{1}{v_1} - \frac{1}{w_1} \quad < \quad \frac{1}{v_2} - \frac{1}{w_2}.$$

The asymptotic behavior of the two-worker bucket brigade depends on how we index the workers given their forward and backward velocities. It was shown in [11] that when the Stability Condition (4.1) holds the system converges to a unique fixed point. Here, we show that if the Stability Condition is strongly violated then the system converges to a chaotic attractor. Before we present this result let us introduce some terminology.

Definition 4.6. *The function f is piecewise smooth [2] if $f(x)$, $f'(x)$, and $f''(x)$ are continuous and bounded for all $x \in [0,1]$ except at a finite number of points.*

Definition 4.7. *The function f is piecewise expanding [2] if $|f'(x)| > 1$, for all $x \in J$, where $J = \{x : x \in [0,1], f'(x)\ exists\}$.*

Definition 4.8. *The points in the set $[0,1] \backslash J$ are called the points of discontinuity [25].*

Theorem 4.1. *Suppose $P(x^k)$ is nonempty for all $k = 0, 1, 2, \ldots$, in Region 4. If the Stability Condition (4.1) is strongly violated, then the map f of the two-worker system is piecewise smooth and piecewise expanding.*

Proof. Since the Stability Condition (4.1) always holds in Region 2, we only need to consider Regions 1, 3, and 4. According to Lemmas 4.3, 4.5, and 4.7, if $1/v_1 - 1/w_1 < 1/v_2 - 1/w_2$ (the Stability Condition is strongly violated) the map f in each of the regions considered is piecewise smooth and piecewise expanding. □

Li and Yorke (see Theorem 1 of [25]) proved that if a function (that maps an interval into itself) is piecewise smooth and piecewise expanding with $d < \infty$ points of discontinuity, then there exist m chaotic attractors, where $m \leq d$. Each chaotic attractor can be described by a density function. Applying Theorem 1 of [25] on the two-worker system leads to the following corollary.

Corollary 4.3. *Suppose $P(x^k)$ is nonempty for all $k = 0, 1, 2, \ldots$, in Region 4. If the Stability Condition (4.1) is strongly violated, then there exist a collection of sets $L_1, \ldots, L_m$ and a set of density functions $\{p_1, \ldots, p_m\}$ such that*

1. *$L_i \bigcap L_j$ contains at most a finite number of points when $i \neq j$;*
2. *each L_i, $1 \leq i \leq m$, is a finite union of disjoint, closed intervals;*
3. *each L_i, $1 \leq i \leq m$, contains at least one point of discontinuity in its interior; thus, if there are d points of discontinuity, then $m \leq d$;*

4. $p_i(x) = 0$ *for* $x \notin L_i$, *and* $p_i(x) > 0$ *for almost all* $x \in L_i$, $1 \le i \le m$;

5. $\int_{L_i} p_i(x)\ dx = 1$ *for* $1 \le i \le m$.

Finally, Theorem 2 of [25] guarantees that the orbit of almost every $x \in [0,1]$ under a piecewise smooth, piecewise expanding map converges to a chaotic attractor. Applying Theorem 2 of [25] on the two-worker system leads to the following corollary.

Corollary 4.4. *Suppose* $P(x^k)$ *is nonempty for all* $k = 0, 1, 2, \ldots$, *in Region 4. If the Stability Condition (4.1) is strongly violated, then for almost every* $x \in [0,1]$, *the orbit of* x *under* f *converges to* L_i *for some* $i \in \{1, \ldots, m\}$.

In Figure 4.2 the Stability Condition (4.1) holds above the diagonal line where the system converges to a unique fixed point. Workers should be indexed such that the system falls into this domain. On the other hand, the Stability Condition is violated below the diagonal line. The system behaves chaotically in this domain and therefore, it should be avoided.

4.6 Transition from stability to chaos

Figure 4.7 shows the orbit diagram [19] of the two-worker system. We fix the values of w_1, v_2, and w_2 at 1, 3, and 2 respectively, and record the orbits under f with different values of v_1. The horizontal axis corresponds to the velocity v_1 and the vertical axis corresponds to the hand-off locations x. For each v_1 we begin from $x^0 = 0$ and plot the subsequent x^k, after 10,000 iterations, vertically above the corresponding v_1. Note that as v_1 increases from 1 to 10 the system moves horizontally from Region 3 to Region 4 in Figure 4.2.

For $1 \le v_1 < v_1^*$, where $v_1^* \equiv (1/v_2 - 1/w_2 + 1/w_1)^{-1} = 1.2$, the system is in the unshaded area of Region 3 in which the Stability Condition (4.1) holds. There is a unique, attracting fixed point x_0^* with $|f'(x_0^*)| < 1$. All orbits converge to the fixed point x_0^*. The fixed point x_0^* becomes neutral when $v_1 = 1.2$ at which $|f'(x_0^*)| = 1$, and the Stability

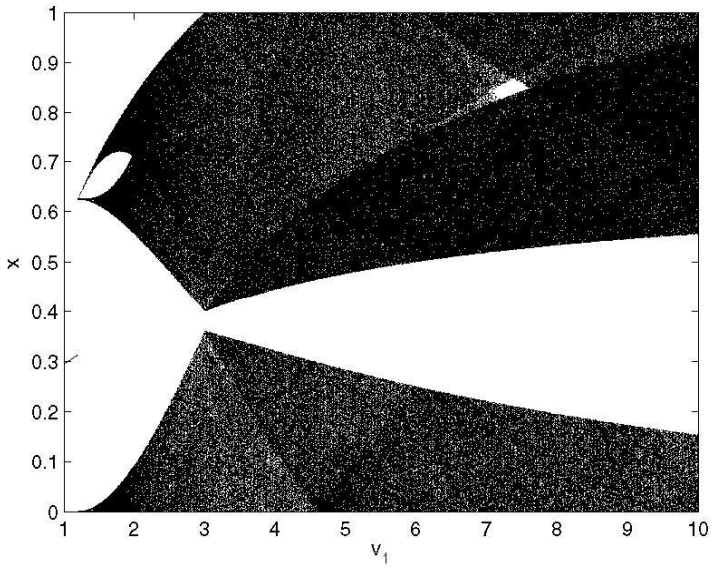

Figure 4.7: **Orbit diagram:** The hand-off locations between the $10,000$th and $11,000$th hand-offs are plotted with different values of v_1 while other velocities are fixed ($w_1 = 1, v_2 = 3$, and $w_2 = 2$).

Condition (4.1) becomes violated. The orbit of $x^0 (\neq x_0^*)$ under f converges to a period-2 orbit (not shown in Figure 4.7). The locations of the period-2 points depend on x^0.

When $v_1 > 1.2$, x_0^* becomes a repelling fixed point because $|f'(x_0^*)| > 1$. According to Corollaries 4.3 and 4.4, an orbit under f converges to a chaotic attractor that consists of a number of disjoint subintervals in $[0, 1]$. Figure 4.8 shows the density distribution of a chaotic attractor under a map f in Region 3 with $v_1 = 1.8, w_1 = 1, v_2 = 3$, and $w_2 = 2$. This map is shown in the bottom graph of Figure 4.5. To create Figure 4.8 the interval $[0, 1]$ is first evenly divided into a number of small, non-intersecting subintervals. The density distribution is then generated by plotting the frequency at which the orbit of $x^0 = 0$ visits each of these small subintervals. Since the function f has only one point of discontinuity (see the bottom graph of Figure 4.5), Condition 3 of Corollary 4.3 leads to the following observation.

Observation 4.3. *For almost every $x^0 \in [0, 1]$, the orbit of x^0 under the map f, shown*

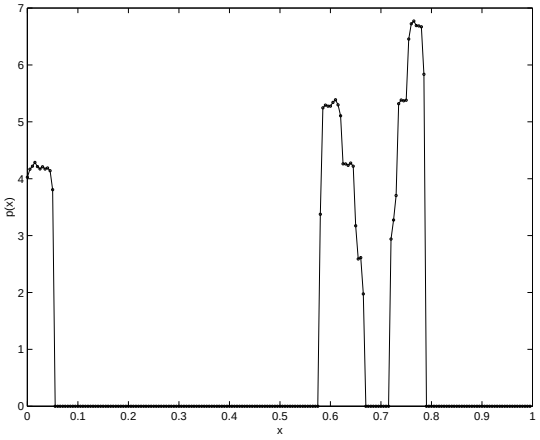

Figure 4.8: The density distribution of the chaotic attractor (generated by frequency plot) under the map shown in the bottom graph of Figure 4.5 (which corresponds to a two-worker system in Region 3)

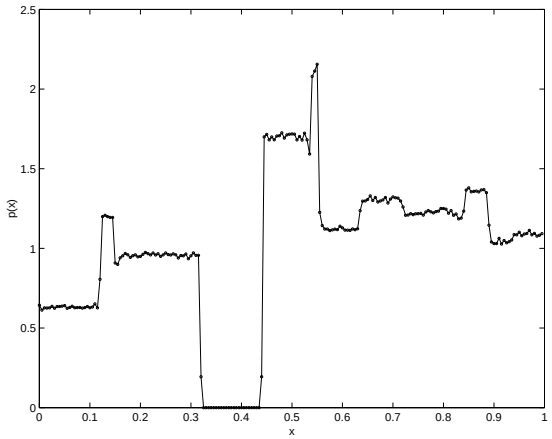

Figure 4.9: The density distribution of the chaotic attractor (generated by frequency plot) under the map shown in Figure 4.6 (which corresponds to a two-worker system in Region 4)

in the bottom graph of Figure 4.5, converges to the same chaotic attractor with the density distribution shown in Figure 4.8.

There is a "white curve" embedded approximately in the range $0.7 \leq x \leq 0.85$ that is not covered by the orbits under f in Figure 4.7. This curve reflects the new repelling fixed point $x_1^* \approx 0.7$ in the bottom graph of Figure 4.5. Furthermore, as stated in Observation 4.2 the hand-off locations in Region 3 are bounded in the range $[0, H_U]$, where $H_U = (1/v_2 + 1/w_2)/(1/v_1 + 1/w_2)$. Note that $H_U \to 1$ as $v_1 \to v_2 = 3$, in agreement with the orbit diagram.

When $v_1 > 3$, we move into Region 4 where the Stability Condition (4.1) is always violated. Figure 4.9 shows the density distribution of a chaotic attractor under a map in Region 4 with $v_1 = 4, w_1 = 1, v_2 = 3$, and $w_2 = 2$. This map is shown in Figure 4.6. The density distribution is generated by frequency plot with $x^0 = 0$. Note that the map f in Figure 4.6 has three points of discontinuity, and all points of discontinuity are contained in the interior of some subintervals where $p > 0$ in Figure 4.9. Condition 3 of Corollary 4.3 leads to the following observation.

Observation 4.4. *For almost every $x^0 \in [0, 1]$, the orbit of x^0 under the map f, shown in Figure 4.6, converges to the same chaotic attractor with the density distribution shown in Figure 4.9.*

4.7 How bad is chaos?

How severely does chaotic behavior affect the performance of a bucket brigade? Is every point in the shaded area in Figure 4.2 "equally chaotic"? To answer these questions we computed the square coefficient of variation (SCV) of inter-completion (or sometimes called inter-departure) times on a grid of points on Figure 4.2. We focus on the inter-completion times because they directly affect the processes downstream from the assembly line. Figure 4.10 shows a surface plot of the SCV of inter-completion times. We begin from

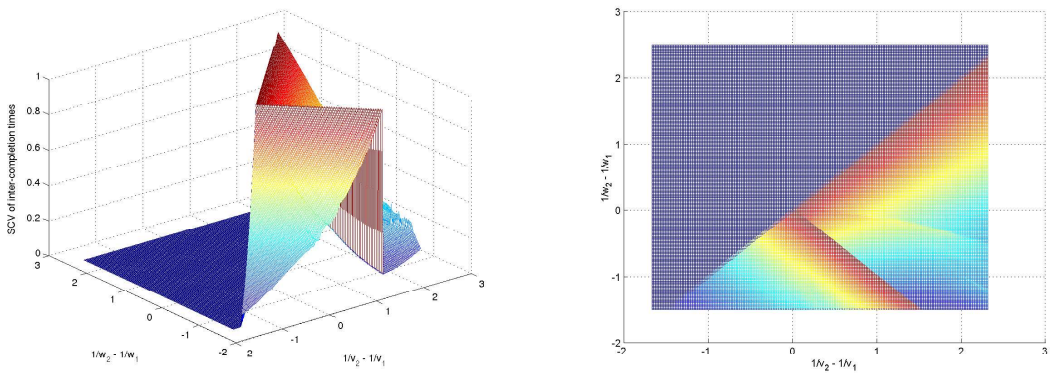

Figure 4.10: A surface plot of the SCV of inter-completion times is viewed from Region 3 to Region 1 (left). The surface plot is viewed from the top (right).

$v_1 = 1/2$, $w_1 = 1/2$, $v_2 = 3$, and $w_2 = 2$, and gradually decrease v_2 and w_2. The SCV is equal to 0 in the domain above the diagonal axis of Figure 4.2 because the system converges to a fixed point and the inter-completion time converges to a constant. However, different areas in the domain below the diagonal axis, where the system behaves chaotically, give rise to different levels of variability. The highest SCV is approximately equal to 0.9, which occurs in the vicinity along the diagonal axis and the axis $1/w_2 - 1/w_1 = -(1/v_2 - 1/v_1)$. The inter-completion time is more sensitive to the hand-off location here in the sense that a slightly different hand-off location may result in a very different inter-completion time. Thus, the system generates high variability in inter-completion time in this vicinity.

We have shown that the two-worker system may behave chaotically and have found where to expect the variability caused by the chaotic dynamics to be severe. Now, we demonstrate that the inter-completion times become more variable when there are more workers on the line. Consider a chaotic bucket brigade assembly line with n workers. Since workers are not blocked, the long-run average throughput of the line is $\sum_{i=1}^{n} 1/(1/v_i + 1/w_i)$. Figure 4.11 shows the SCV of inter-completion times as a function of n based on simulations with $v_i = n - 0.1 \times (i-1), w_i = 3$, for $i = 1, \ldots, n$. The SCV increases with n. This suggests that every time we add a worker into an already chaotic line, more variability is induced by the dynamics of the system. The negative effect of the chaotic dynamics

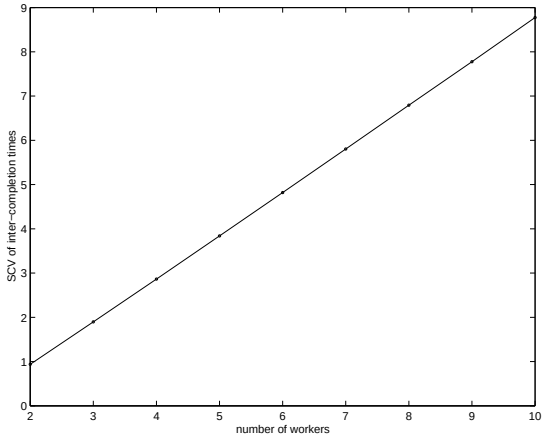

Figure 4.11: The SCV of inter-completion times increases with the number of workers in a chaotic bucket brigade assembly line.

becomes more prominent when we have more workers on the line.

How does a bucket brigade assembly line that is deterministic but chaotic compare to an assembly line that is truly random? Consider the performance of the same group of workers on another assembly line in which each worker works independently from others. Let the time to complete an item by worker i be exponentially distributed with rate $1/(1/v_i + 1/w_i)$. Also assume that when a worker finishes an item, it takes negligible time to start a new item. The expected throughput of this stochastic assembly line is equal to the long-run average throughput of the chaotic bucket brigade. Due to the memoryless property of the exponential distribution, the inter-completion times of the stochastic line are also exponentially distributed with rate $\sum_{i=1}^{n} 1/(1/v_i + 1/w_i)$ and the SCV of inter-completion times of the stochastic line is 1. Comparing this result with the example in Figure 4.11, we see that the variability of the chaotic bucket brigade is greater than that of the stochastic line when $n \geq 3$. This demonstrates that a deterministic manufacturing system under an improper configuration may have higher variability of output than that of a random system in which products cannot remember the work invested in them!

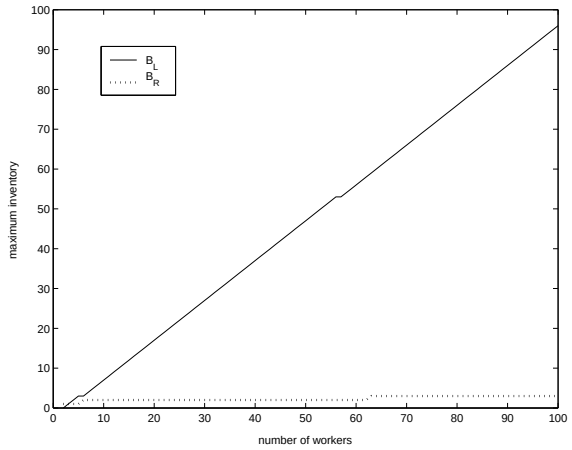

Figure 4.12: The maximum inventory in both buffers increases with n, however, the maximum inventory in B_L increases with a much higher rate.

To illustrate the corrupting influence of an improperly configured bucket brigade, consider an assembly system in which two assembly lines L and R merge at a point O. Each line produces a subcomponent and completed instances of these subcomponents are combined to produce the final product at O. A buffer B_L (B_R) is located at the end of line L (R). When an item from line L (R) is completed it is combined with an item from buffer B_R (B_L) if B_R (B_L) is not empty. Otherwise, it is stored in buffer B_L (B_R). Assume that each line runs a bucket brigade with n workers and the team on line L is identical to the team on line R. Furthermore, assume that Stability Condition (4.1) is violated on line L but it holds on line R. Figure 4.12 shows the maximum inventory in each buffer as a function of n. For each n we perform a simulation with 10,000 final items produced and the maximum inventory of each buffer is recorded. The maximum inventory in both buffers increases with n, however, the maximum inventory in B_L increases with a much higher rate. This is due to the fact that the chaotic bucket brigade on line L can produce many items in a very short time while buffer B_R is empty. As a result, the inventory in B_L can reach a high level before it is cleared. Apparently, this can be avoided and no buffers are needed if the bucket brigade on line L is configured properly and the two lines are synchronized.

4.8 Summary

From the management point of view we want the Stability Condition (4.1) to hold so that the system converges to a fixed point with regular production. We can achieve this by indexing the workers in the correct order. The Stability Condition (4.1) can be interpreted as: *the worker who is less slowed by work should be downstream.*

An interesting property of this generalized model is the capacity of exhibiting chaotic behavior. Even in the simplest case with only two workers, the dynamics of the system can be nontrivial and may induce large variability. This demonstrates that the variability caused by the dynamics of a purely deterministic system can be so large that the output is effectively random. Different areas in the chaotic domain in Figure 4.2 give rise to different levels of variability. Furthermore, the effects of the chaotic dynamics become more prominent when there are more workers on the line.

Figure 4.2 summarizes the dynamics of the two-worker system. Given the velocities of the workers, we can tell the dynamics and predict the asymptotic behavior of the system by identifying which part of the figure the system is located. We even know how the asymptotic behavior changes as the system shifts from one part of the figure to another when the velocities change (which may be due to learning effect).

If, for some reasons, the system cannot satisfy the Stability Condition and chaotic behavior is inevitable, then Figures 4.8 and 4.9 are useful in designing a bucket brigade assembly line. They suggest that there exist some subintervals on the assembly line where hand-offs will never occur (or occur with very low density). On the other hand, there exist some subintervals with very high density. We can design the assembly line so that the work content that does not facilitate hand-offs falls within those subintervals with zero or low density, but not those with high density.

Some interesting questions remain. Where exactly is a chaotic attractor located on an assembly line? Can the density functions shown in Figures 4.8 and 4.9 be determined in closed form? How can we recognize chaotic behavior on an assembly line in real life?

Chapter 5

Bucket Brigades with Multiple Job Sources: A Case Study

The bucket brigade protocol can be adapted to an order-picking operation where jobs arrive arbitrarily in time and are introduced into the system at various points along the flow line. We perform computer simulations based on data from a distribution center and compare the adapted bucket brigade protocol with the current order-picking practice at the distribution center.

5.1 Introduction

Order-picking is very labor intensive and it typically accounts for significant operating cost within a warehouse. As a result, the management of a warehouse often wants to fully utilize its picking capacity while ensuring the speed and accuracy of delivery. This leads to the question of how to dispatch order pickers in the warehouse so that the order-picking operates in the most efficient and cost effective way.

Many warehouses adopt the *zone-picking* protocol in which every worker is restricted to a rigid section of storage area in a warehouse. This approach cannot achieve effective sharing of work among workers (thus, it does not fully utilize the warehouse's picking

capacity) because it does not facilitate workers from different sections to help each other. Furthermore, given the vicissitudes of demand it is impossible to balance the work load on each worker in a timely manner. Thus, under zone-picking different workers are generally not utilized to the same level.

Recently, Bartholdi and Eisenstein introduced the *bucket brigade* protocol [9, 12] for order-picking in which each worker is allowed to move to any pick location. This protocol facilitates workers sharing the work load and has been shown to be more effective than zone-picking in many circumstances [9, 12]. A comparison of bucket brigades and zone-picking can be found in [12, 13].

Here, we study a distribution center to investigate the potential of bucket brigades for its order-picking operation. In this particular case, customer orders arrive randomly in time and are released into the system at various locations along a flow line. Besides fully utilizing its picking capacity, the distribution center has other goals such as minimizing both the time to fulfill an order and the work to consolidate orders at downstream from order-picking. We adapt the bucket brigade protocol to this environment and compare the performance of different configurations of bucket brigades with zone-picking by running computer simulations based on daily data.

5.2 Order-picking at the distribution center

The distribution center studied here is a leading worldwide distributor of tools and service parts. It supplies products to industrial and large commercial facilities worldwide. The company has been in business for more than ninety-five years and specializes in quick and accurate delivery.

Customer orders are received at the distribution center via phone calls, e-mails, and its business website twenty-four hours a day, seven days a week. Orders are picked between 8:30 AM and 6:20 PM during an operational day. Orders that arrive after the operational hours are carried over to the next operational day. On the other hand, orders that arrive

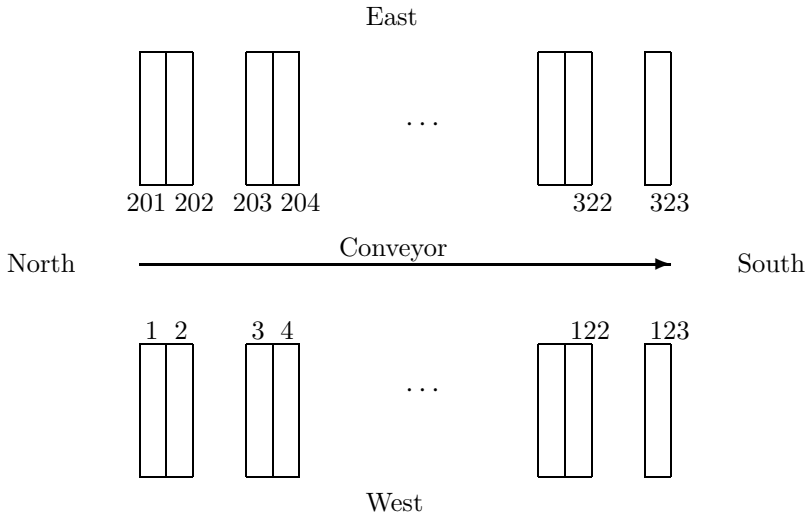

Figure 5.1: The layout of an aisle (not to scale)

during the operational hours (between 8:30 AM and 6:20 PM) are released every five minutes for picking to ensure quick response and therefore, are not batched with many other orders. Furthermore, customer orders are typically small (with an average of less than three items per order). As a result the order-picking at the distribution center is a *low-density* order-picking operation — workers must walk considerable distance between picks.

The pick module of the distribution center for broken-case picking consists of three levels. We call each level an *aisle*. The layout of an aisle is shown in Figure 5.1. Each aisle operates independently from others and this allows us to focus our study on a single aisle. On each aisle a central conveyor runs from North to South, splitting the aisle into two sides. There are *rows* emanating from each side of the conveyor. Each row consists of a sequence of bin shelves, which stand side by side in a row perpendicular to the conveyor. The rows are numbered from North to South ranging from 1 to 123 on the West side, and ranging from 201 to 323 on the East side. The rows are aligned so that there is a *sub-aisle* in every other row. These sub-aisles are for workers to retrieve and restock stock-keeping-units (SKUs) on the bin shelves. Furthermore, each row is partitioned into sections of

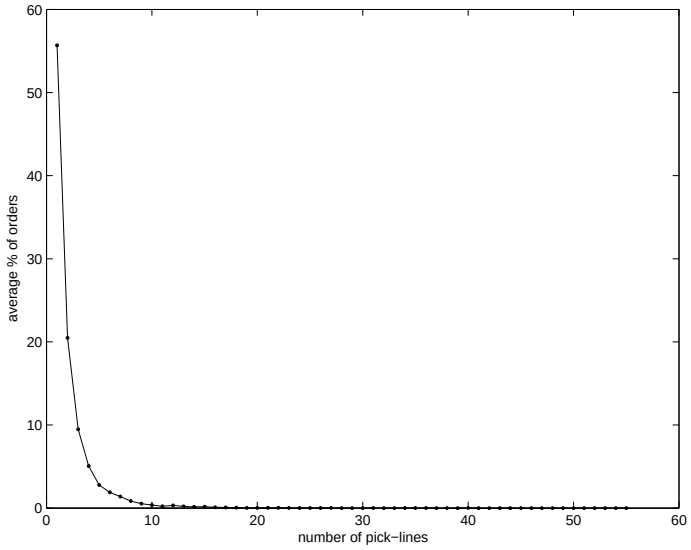

Figure 5.2: The number of pick-lines per order is typically small. Eighty percent of orders contain only 3 pick-lines or fewer. The average is 2.18 pick-lines per order and the median is 1 pick-line per order.

shelving. The sections are numbered sequentially from 1 (nearest to the conveyor) to 20 (farthest from the conveyor).

An *order* is a list of SKUs to be picked for a single customer. Each SKU is represented by a *pick-line*, which includes the location of the SKU on the aisle. Multiple pick-lines are used if more than one piece is requested for a SKU. Orders are released for picking through printers located along each side of the conveyor. Figure 5.2 shows a typical distribution of orders on an aisle over different numbers of pick-lines. As shown in Figure 5.2 customer orders are typically "small". Eighty percent of orders contain only 3 pick-lines or fewer. The average is 2.18 pick-lines per order and the median is 1 pick-line per order.

Figure 5.3 shows the distribution of demand over time. The demand peaks, on average, at 9:00 AM and 2:00 PM during an operational day. Note that customer orders arrive twenty-four hours a day even though the order-picking operates only about ten hours a day. To ensure quick delivery customer orders are batched and released for picking every five minutes within the ten hours of operation.

Figure 5.4 shows the average distribution of pick-lines for a day over an aisle. Since

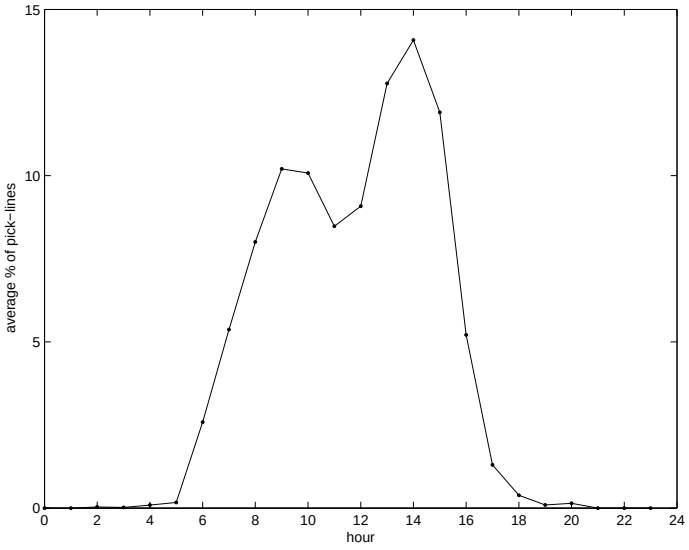

Figure 5.3: The average demand peaks at 9:00 AM and 2:00 PM during a day.

SKUs are stored along each side of the conveyor according to product type (for example, all drill bits are located together) the distribution of pick-lines along each side of the conveyor is, in general, not uniform. On the other hand, SKUs within a row are stored according to popularity. That is, the most popular SKUs are stored close to the start of the row. Figure 5.5 shows the average distribution of pick-lines for a day along a row. Eighty percent of picks within a row are from the first 7 sections of shelving. On average, a worker travels only about $4/20 = 20\%$ of the length of a row. The median travel distance into a row is 3 sections of shelving.

5.3 Issues faced by the distribution center

The distribution center currently adopts a zone-picking protocol for the order-picking operation on each aisle. Each side of the conveyor is partitioned into three contiguous sections called *zones*, such as shown in Figure 5.6. A printer is installed at the center of each zone and it prints pick-lines only for SKUs located within the zone. An order that requests SKUs in different zones is split into several *pick-lists*, each of which is printed by one printer. Pick-lists for the same order are released simultaneously and are picked

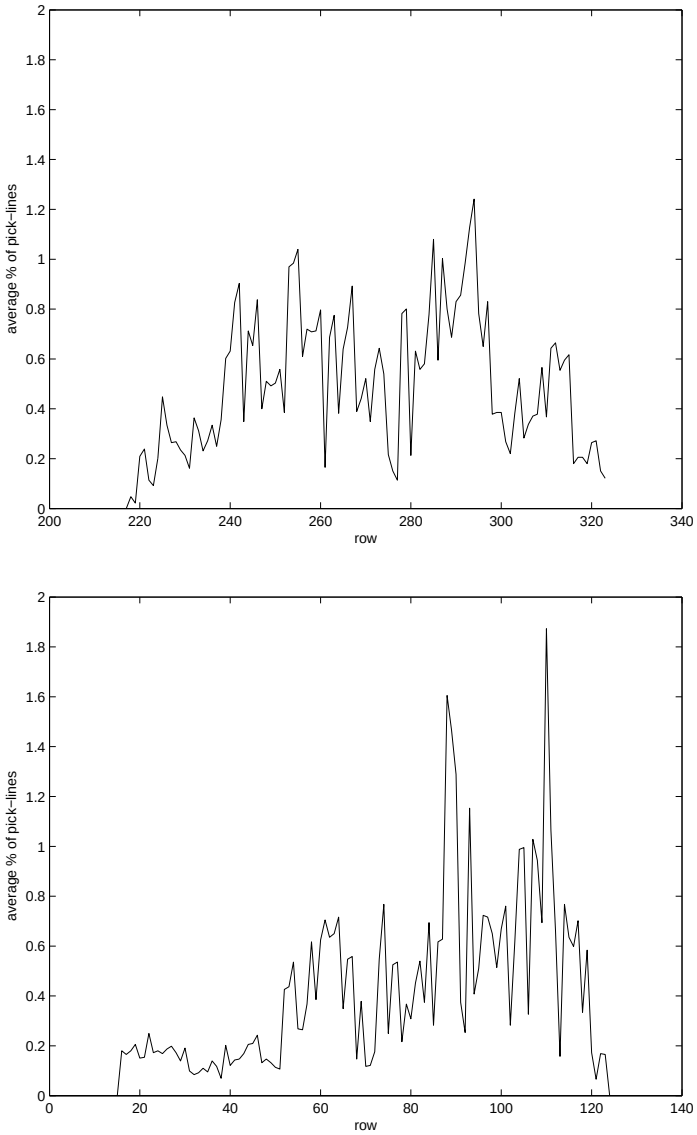

Figure 5.4: The average distribution of pick-lines for a day on the East side (top) and the West side (bottom) of an aisle

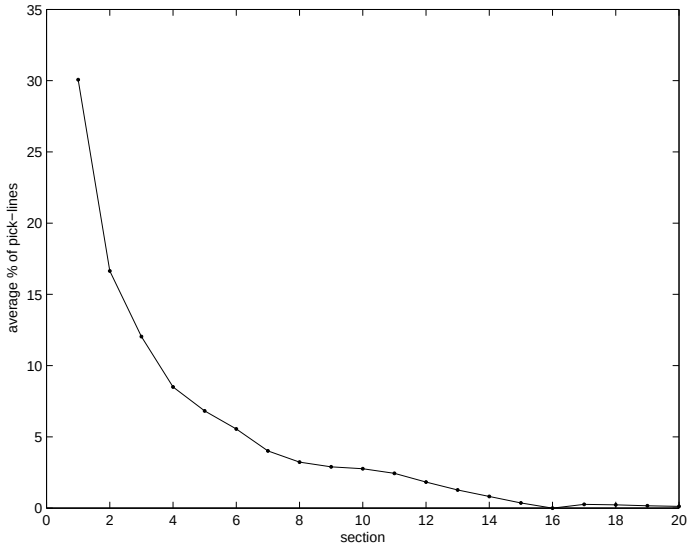

Figure 5.5: Eighty percent of picks within a row are from the first 7 sections of shelving. On average, a worker travels only about $4/20 = 20\%$ of the length of a row. The median of travel distance into a row is 3 sections of shelving.

independently. When all pick-lists for an order have been picked they are consolidated in a separate area, downstream from order-picking.

Each worker is assigned to a zone and is responsible for picks within the zone. However, a zone may be staffed with several workers. When there are pick-lists available at a printer a worker takes several of them and proceeds to pick the SKUs on the pick-lists. The worker puts the retrieved items into totes, which he slides along the passive lane of the conveyor while "assembling" his pick-lists. When a pick-list is completed the worker puts the tote(s) containing its retrieved items onto the powered portion of the conveyor, which brings them to a separate area for consolidating and then shipping. When the worker finishes all his pick-lists he returns to the printer for more work. If there are no pick-lists available at the printer the worker becomes idle and he may help his colleagues in other zones in some ad hoc manner, or he may use this time to do restocking or cleaning.

The management of the distribution center is concerned with the following issues.

1. **Cycle time:** The most important concern of the distribution center is the *cycle time* of an order, which is the duration between the time when the order arrives

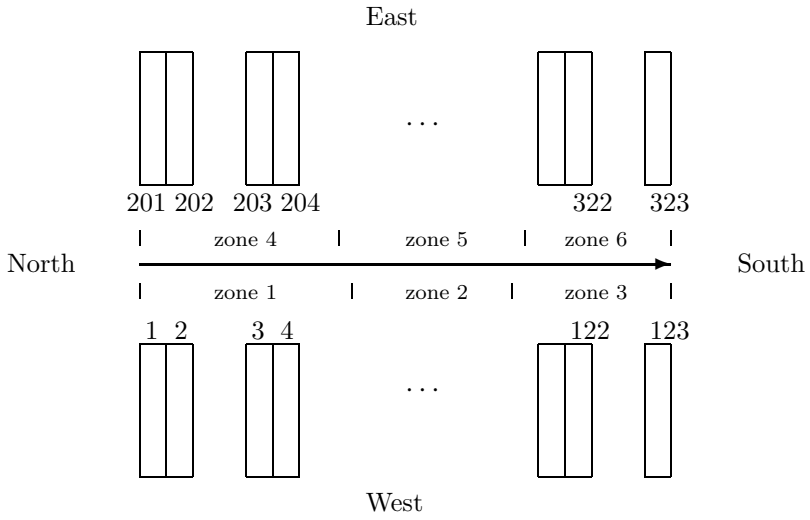

Figure 5.6: Under zone-picking each worker picks SKUs located within his assigned zone. A printer is located at the center of each zone to print pick-lists.

and the time when it is completed and ready for shipping. The managers of the distribution center strive to reduce the average cycle time per order as much as possible to ensure quick delivery.

2. **Labor in order-picking:** The distribution center also keeps the labor involved in order-picking as low as possible. Currently, there are 14, 9, and 9 workers in the first, second, and third aisles respectively. The management would like to fully utilize the current capacity in order-picking.

3. **Work in sortation and consolidation:** Another concern is to minimize the work to sort and consolidate any split orders. Sortation and consolidation are done in a separate area of the warehouse, downstream from order-picking. Minimizing the work in sortation and consolidation will reduce the average cycle time of an order.

Apparently, the first two issues conflict with each other. Increasing the labor in order-picking will help shorten the average cycle time of an order. However, the management would like to keep the cost of labor in order-picking low. On the other hand, reducing the work in sortation and consolidation will help shorten the average cycle time.

5.4 Order-picking by bucket brigades

We investigate the potential of bucket brigades in order-picking at the distribution center with respect to the issues discussed in the previous section: (1) average cycle time per order; (2) labor in order-picking; and (3) work in sorting and consolidating split orders.

The model introduced in [11] and further discussed in Chapter 4 is suitable for the order-picking operation at the distribution center because workers can pass each other on an aisle in the distribution center. Furthermore, in the low-density order-picking at the distribution center the time to walk back to get more work is not significantly less than the time to work forward to pick a few SKUs. We adapt the model discussed in Chapter 4 to the case of the distribution center.

5.4.1 How to form bucket brigades?

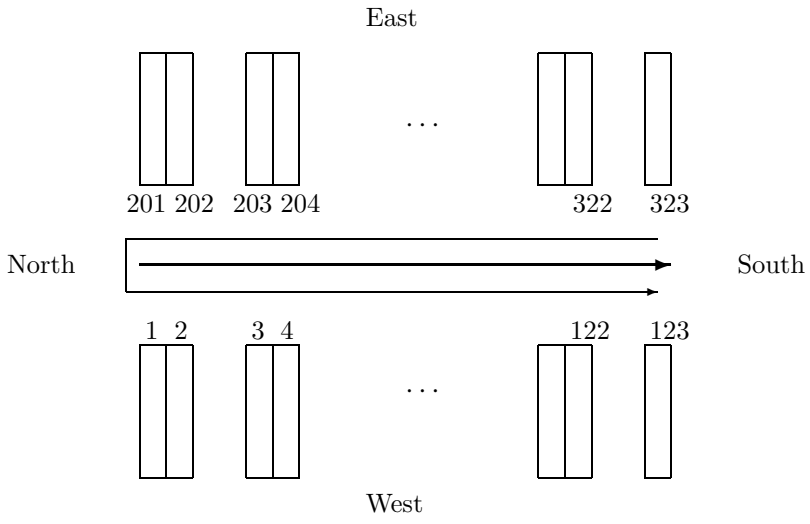

Figure 5.7: A single bucket brigade is formed across the aisle. No orders will be split as orders are progressively assembled by workers along the entire flow line.

We first form a single bucket brigade across the entire aisle. The flow line is illustrated in Figure 5.7. The line begins from row 323 (East side of the aisle), passes through row 201,

continues on row 1 (West side of the aisle), and terminates at row 123. The flow line is a "U-shaped" line that runs around the central conveyor. An advantage of forming a single bucket brigade is that each order can be progressively assembled by workers along the flow line, which covers the entire aisle. Thus, no orders will be split within an aisle. This eliminates the work to sort and consolidate split orders within an aisle (though a customer order may be split among aisles because aisles operate independently). However, as shown in Figure 5.7, each of the orders that request SKUs from both sides of the conveyor must be carried at least 10 additional feet from one side to the other side of the conveyor. This extra travel increases the time to complete each order.

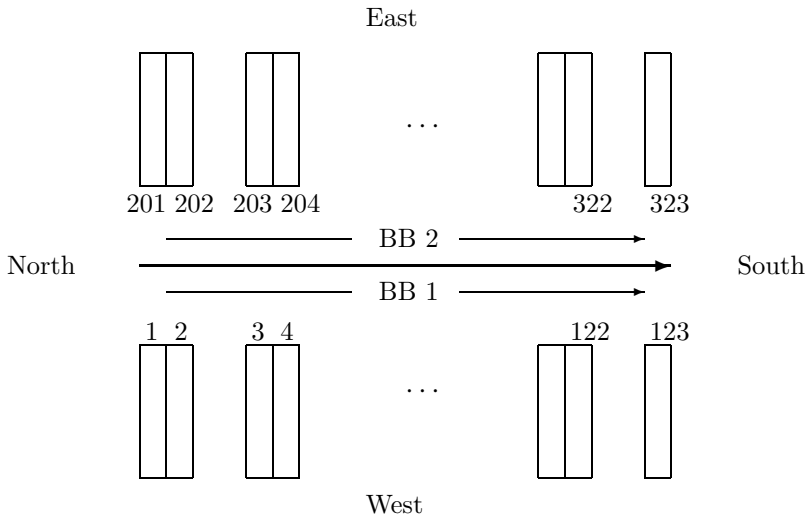

Figure 5.8: Two independent bucket brigades are formed across the aisle. An order requesting SKUs from both sides of the conveyor will be split into two pick-lists, each of which will be picked by a bucket brigade.

To avoid the extra travel incurred in the previous configuration one can run two independent bucket brigades, each of which covers one side of the conveyor. Figure 5.8 shows the configuration with two bucket brigades on an aisle. Let BB 1 and BB 2 denote the bucket brigades on the West and East sides of the aisle respectively. Since these bucket brigades operate independently an order requesting SKUs from both sides of the conveyor

will be split into two pick-lists, each of which will be picked by a bucket brigade.

Since the pick density is low, workers walk considerable distance between picks. In other words, workers travel through significant "dead space" along the aisle. A worker picks up pick-lists from a printer and proceeds to the first SKU to be picked. The distance between the printer and the first pick is considered as dead space. Furthermore, the distance between two successive picks is also considered as dead space. Zone-picking reduces dead space by splitting each order into multiple pick-lists. However, this incurs extra work in sortation and consolidation. On the other hand, running bucket brigades reduces the work in sortation and consolidation but incurs more dead space in the order-picking operation. Thus, the question is how to trade off the two conflicting factors: dead space in the order-picking operation versus work in sortation and consolidation.

We will run one and two bucket brigades over the entire aisle and compare their performance with zone-picking through computer simulations based on the customer order data from the distribution center.

5.4.2 Where to release orders?

When a single bucket brigade is formed across the entire aisle no orders will be split. Thus, a pick-list contains all the SKUs requested by an order. SKUs on a pick-list are picked successively in a sequence according to the direction of the flow line shown in Figure 5.7.

Pick-lists are released through printers located along each side of the conveyor. Customer orders for the distribution center are very variable in quantity and location. Some orders can be completed within the first few rows, while others require a complete traversal of the aisle. Similarly, the variability in orders means that some orders require picking within the first few rows, while others have no picks until the last few rows.

To reduce dead space, a pick-list is released through the *nearest* printer *upstream* from the first pick of the pick-list. When a worker walks back to get more work, he takes the first work he encounters. The first work could be either the work carried by his co-worker,

or paper pick-lists at a printer. This scheme helps reduce travel, because workers do not have to walk all the way back to the start of the line to get more work and then have to walk forward to his first pick.

For the configuration with two bucket brigades on an aisle, a pick-list covers only SKUs on one side of the conveyor. SKUs on a pick-list are picked successively in a sequence according to the directions of flow lines shown in Figure 5.8. Similar to the previous configuration, a pick-list is released through the nearest printer upstream from the first pick of the pick-list.

5.4.3 Primary printer assignment

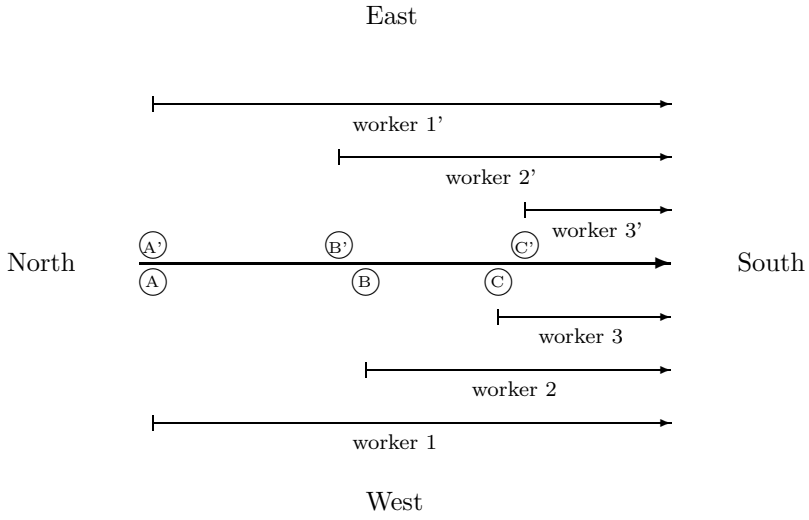

Figure 5.9: Each worker in a bucket brigade flow line is assigned a primary printer. A worker picks only SKUs downstream from his primary printer.

The bucket brigade protocol proposed in [11] requires each worker (after relinquishing his work) to walk back until he encounters a co-worker or reaches the start of the line. Since orders are batched and released every five minutes, workers may become idle when all orders (for a particular batch) are picked before the release of the next batch of orders. Consequently, any workers may wait beside the printer at the start of the line for the next

batch of pick-lists. (The wait is at most five minutes long because pick-lists are released every five minutes.) New pick-lists may only emerge from a printer located downstream from the workers. Then, workers must walk forward to the printer to get new work. This creates unnecessary travel.

A way to overcome this problem is to "station" some workers at each printer when the system is idle so that when new pick-lists emerge from a printer, there will be some workers to pick them right away. We assign a *primary printer* to each worker. The primary printer of a worker is the first printer that the worker is responsible for picking up pick-lists along the flow line. In other words, a worker is only responsible for any SKUs downstream from his primary printer. When a worker is idle he will wait at his primary printer.

Figure 5.9 shows an example of two bucket brigades formed on an aisle. Each bucket brigade is formed on one side of the conveyor and contains only three workers. There are three printers located along each side of the conveyor. In this example a printer is assigned arbitrarily as primary printer to each worker. For instance, printer A on the West is assigned to worker 1, and printer B' on the East is assigned to worker 2'. Note that a printer can be assigned as primary printer to more than one worker on a bucket brigade flow line (although each printer is assigned to exactly one worker in this example).

5.4.4 Extended bucket brigade protocol

Since pick-lists are released into a bucket brigade flow line through multiple printers, we need to extend the bucket brigade protocol in [11] to cope with this additional complication. Furthermore, at any instant in time a worker may carry some unpicked pick-lines and/or some retrieved *physical* items. An unpicked pick-line and a retrieved physical item are indistinguishable when workers follow the extended protocol. Thus, in the following description an *item* refers to either an unpicked pick-line or a retrieved physical item.

We index workers in a bucket brigade from 1 to n. Each worker i in the bucket brigade follows the rules stated in Table 5.1. Under these rules workers are not restricted to a

Table 5.1: The Bucket Brigade Rules determine what each worker i should do. Under these revised rules workers are not restricted to a fixed sequence along the line.

Forward Rule:

- When you carry some items, work forward until one of the following cases occurs:
  1. Your items are handed off to a co-worker, in which case follow the **Backward Rule**.
  2. You have completed and put all your items on the powered conveyor, in which case follow the **Backward Rule**.
- When you do not carry any item, go forward until one of the following cases occurs:
  1. There are no pick-lists available at any of the printers ahead of you, in which case follow the **Backward Rule**.
  2. You meet a worker $k > i$ walking back, in which case follow the **Backward Rule**.
  3. You reach a printer where there is at least one pick-list available, in which case pick up the pick-list(s) and follow the **Forward Rule**.

Backward Rule:

- Walk back until one of the following cases occurs:
  1. You encounter a worker $j < i$ working forward with items, in which case take over his items and follow the **Forward Rule**.
  2. You encounter a worker $j < i$ going forward without items, in which case follow the **Forward Rule**.
  3. You reach a printer where there is at least one pick-list available, in which case pick up the pick-list(s) and follow the **Forward Rule**.
  4. You reach your primary printer and there are no pick-lists there, in which case wait until there are pick-lists available from your primary printer or from any of the printers downstream from you, then follow the **Forward Rule**.

fixed sequence along the line. We assume that workers are aware of the availability of pick-lists at each printer. This can be accomplished by installing a light at each printer that is turned on when there is at least one pick-list at the printer.

5.5 Performance measures

Some aspects of the problem faced by the distribution center can be captured by measuring the following variables in the computer simulations. These variables reflect (to a certain degree) the issues described in Section 5.3.

1. **Flow time:** Define the *flow time* of an order as the duration between the moment when the order is released to the distribution center and the moment when its last SKU is picked and put onto the powered conveyor. This does not include the travel time to the sorting and consolidating area, the time to sort and consolidate picked items if the order is split, and the travel time to the shipping department (all these measures are beyond the scope of our simulations). If an order is not split, minimizing its flow time is almost equivalent to minimizing its cycle time (neglecting the effect of travel time to downstream from order-picking).

2. **Travel by workers:** Workers walk considerable distance between picks in the distribution center. The more travel involved in order-picking the more labor we need to ensure quick delivery. As a result, travel must be reduced to keep the labor involved in order-picking low.

3. **Work-in-process (WIP):** Define the WIP on an aisle at any instant in time as the number of incomplete orders on the aisle. This includes the paper pick-lists (new orders) at the printers as well as orders being assembled by the workers. We want the WIP level as low as possible because high WIP level induces more errors, which delay the departure of orders.

4. **Work-in-consolidation (WIC):** If two bucket brigades or zone-picking are run on an aisle, some orders will be split into several pick-lists and different pick-lists are picked independently. When all the SKUs on the same pick-list are picked they are carried by the powered conveyor to an area for consolidation. These SKUs are stored in that area until they are consolidated and shipped with all other SKUs from other pick-lists for the same order. Define the WIC of an aisle at any instant in time as the number of retrieved items (retrieved pick-lines) from the aisle that are stored in the consolidation area. The WIC level estimates the space and labor required to sort and consolidate split orders.

5.6 Effects of number of printers

The dead space incurred in order-picking may be reduced if more printers are distributed across the aisle. This is because pick-lists are released, on average, at locations closer to their SKUs if more printers are installed and are properly located on the aisle. This helps reduce the time to complete a pick-list. Thus, a way to reduce the average flow time of an order is to install more printers across the aisle.

We examine the effects of the number of printers on an aisle, through simulations, on the following three order-picking policies:

1. one bucket brigade across the entire aisle (see Figure 5.7);
2. two bucket brigades, each on one side of the conveyor (see Figure 5.8);
3. zone-picking, each side of the conveyor is covered by an equal number of zones (see Figure 5.6).

There are 14 workers on the aisle (this is the number of workers on the first aisle in the distribution center currently). We make the following assumption on the workers in the simulations.

Assumption 5.1. *Each worker i has a constant work (forward) velocity v_i and a constant walk-back (backward) velocity w_i.*

We set the walk-back velocity of each worker to be 3 feet/second. To capture the difference in experience of workers, we set the work velocity (in feet/second) of a worker to be a constant randomly drawn from the interval [3, 3.6]. The work velocity of a worker is set to be slightly higher than his walk-back velocity because, according to the management of the distribution center, workers like to spend more time (to have some break) to walk back for more work.

We also make the following assumption on the time spent by a worker in a sub-aisle to retrieve an item.

Assumption 5.2. *It takes a constant amount of time to retrieve an SKU from a sub-aisle.*

Based on a simple survey in the distribution center, this duration is set to be 8.8 seconds.

All workers belong to the same team when a single bucket brigade is formed across the aisle. When two bucket brigades are formed on the aisle, each brigade has 7 workers. Workers in a bucket brigade with n workers are indexed so that $v_1 \leq v_2 \leq \ldots \leq v_n$. Within a bucket brigade each printer could serve as the primary printer of several workers. Workers are assigned as evenly as possible to the printers. Furthermore, workers with higher index are assigned to printers downstream of the line. Similarly, for zone-picking workers are assigned as evenly as possible to the zones. We assume that each time a worker takes as many pick-lists as possible (as long as they are available) up to b pick-lists from a printer. Here, we set $b = 6$ and we will investigate the effects when b is set to other values in the next section.

Figure 5.10 shows the average flow time per order (in minutes), under different order-picking policies, with different numbers of printers on the aisle. The number of printers varies from 2 to 14 (which equals the number of workers on the aisle). For each number of printers, we enumerate all possible printer locations for each policy and select the set

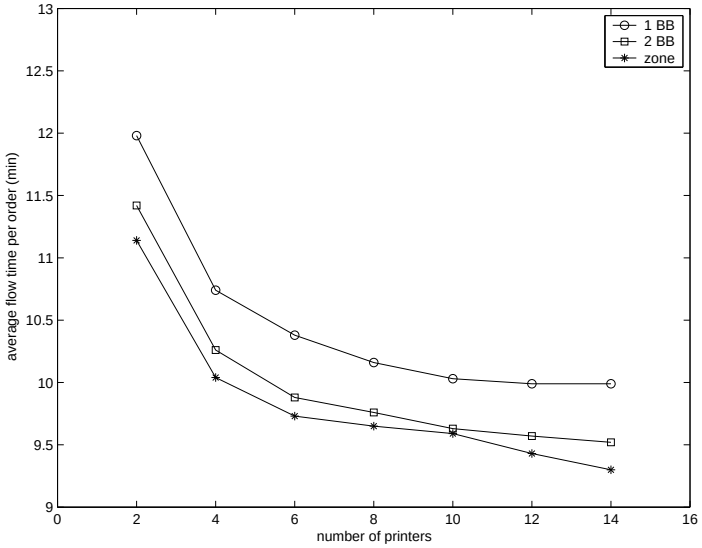

Figure 5.10: The average flow time of an order decreases with the number of printers on the aisle.

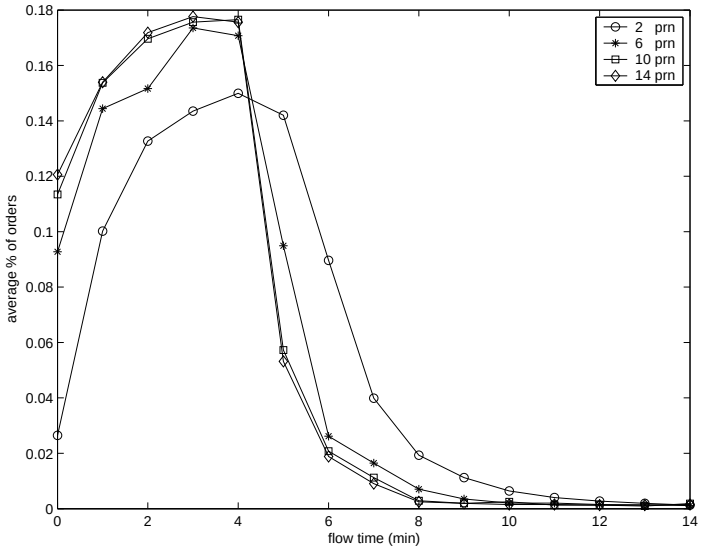

Figure 5.11: The mean and variance of flow time per order decrease with the number of printers.

of printer locations that gives the lowest average flow time per order. This set of printer locations could be different for different policies. Apparently, for each policy, the average flow time per order decreases as the number of printers on the aisle increases. For a given number of printers, zone-picking gives the lowest average flow time among all the policies, followed by the policy with two bucket brigades. Figure 5.11 shows the flow time distributions with different numbers of printers under the policy of single bucket brigade. The mean and variance of flow time per order decrease with the number of printers. Similar results are found for two bucket brigades and zone-picking.

Zone-picking outperforms bucket brigades with respect to flow time because orders are split (if necessary) into several pick-lists, each of which is picked independently from others. This allows workers in different zones to pick the SKUs for each split order *in parallel*. However, splitting the orders causes more work in sortation and consolidation, which delays the departure of orders. On the other hand, the single bucket brigade running across the entire aisle progressively assembles each order. This requires no orders split within the aisle. However, it takes a longer time to complete a pick-list (order) because SKUs at downstream of the line are picked only after all requested SKUs at upstream have been picked.

As shown in Figure 5.10 the performance of the policy of two bucket brigades lies between the above two policies. It gives lower average flow time than the single bucket brigade but requires some orders to be split into two pick-lists, each of which is picked on one side of the conveyor. Figure 5.12 compares the percentage of split orders by two bucket brigades with zone-picking when 14 printers are used on the aisle. As expected more orders remain non-split when two bucket brigades are run on the aisle. Thus, running two bucket brigades serves as a compromise between the single bucket brigade and zone-picking. It gives lower average flow time than the single bucket brigade and splits fewer orders than zone-picking.

Figures 5.13 and 5.14 show the average travel per order per worker (in feet) and the av-

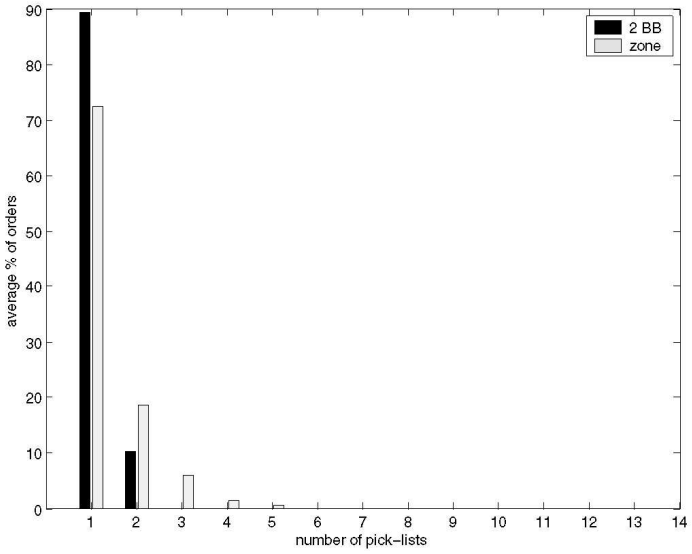

Figure 5.12: As expected zone-picking splits more orders than the policy of two bucket brigades.

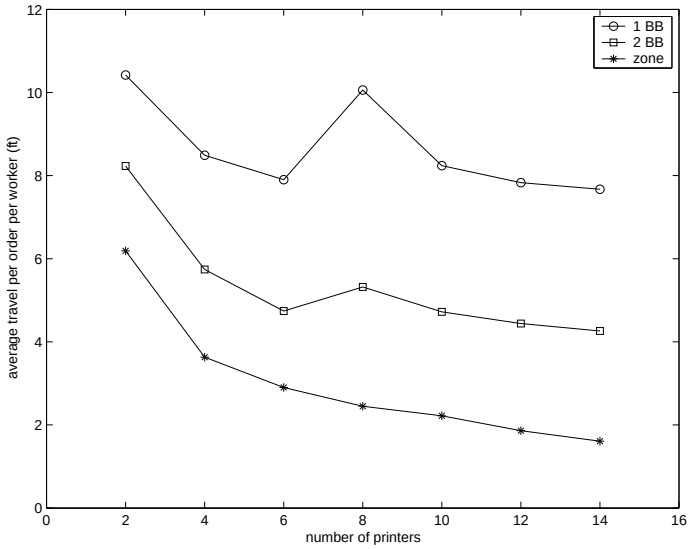

Figure 5.13: The average travel per order of a worker decreases with the number of printers for zone-picking. However, it suddenly increases when the number of printers reaches 8 and then continues to decrease for bucket brigades.

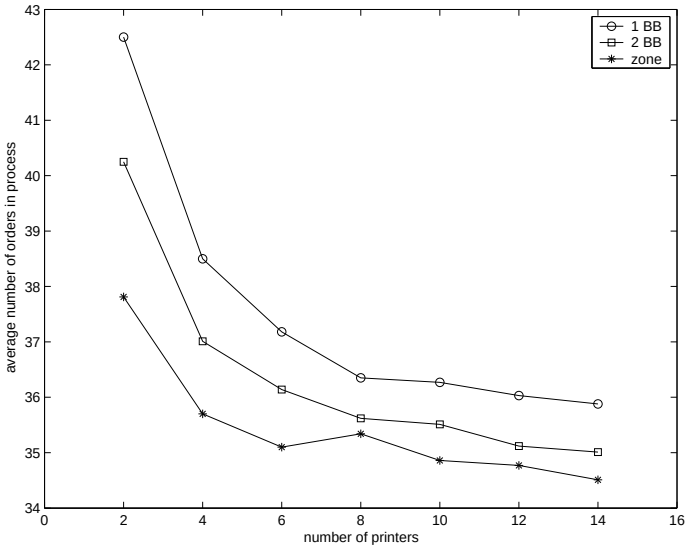

Figure 5.14: The average WIP on the aisle decreases (with some small fluctuation) with the number of printers on the aisle.

erage WIP on the aisle (in orders) respectively, under different order-picking policies, with different numbers of printers on the aisle. The same set of printer locations for Figure 5.10 is used for each order-picking policy and each number of printers. For zone-picking the average travel decreases with the number of printers. This confirms our previous argument that dead space may be reduced when more printers are installed and are properly located across the aisle. However, for bucket brigades, the average travel first decreases with the number of printers. It then suddenly increases when the number of printers reaches 8 but continues to decrease as the number of printers gets larger. The sudden increase in travel is due to the bucket brigade protocol, which requires idling workers to walk forward to printers where new pick-lists are emerging. Thus, more sources of pick-lists (printers) may induce more travel for idling workers. Among all policies zone-picking gives the lowest average travel, whereas the single bucket brigade requires the most travel. The average travel by two bucket brigades lies between the above two policies.

The behavior of the average WIP under each order-picking policy is, in general, similar to that of the average flow time. Among all policies zone-picking always results in the lowest average WIP on the aisle for a given number of printers. This could be misleading

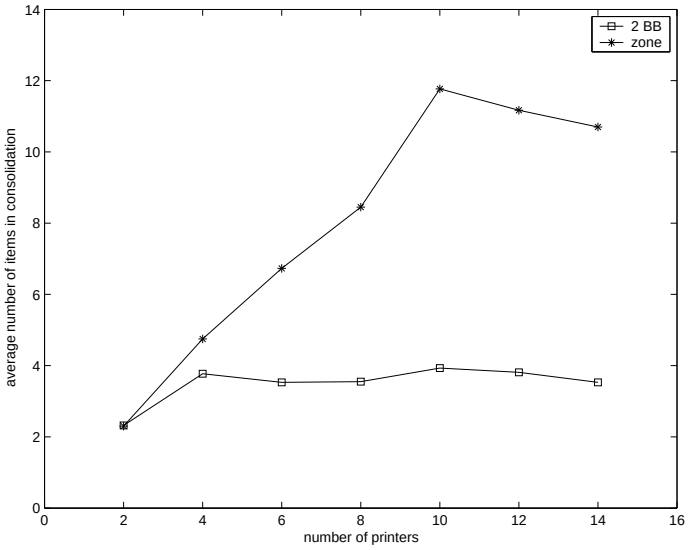

Figure 5.15: Zone-picking results in higher average WIC than the policy of two bucket brigades.

because even though it gives the lowest WIP on the aisle, zone-picking does accumulate *more* WIC than bucket brigades! Figure 5.15 shows the average WIC (in items) of an aisle by two bucket brigades and zone-picking. For two bucket brigades the average WIC remains more or less constant as the number of printers on the aisle increases. This is because the number of split orders does not increase with the number of printers for the policy of two bucket brigades. However, for zone-picking the average WIC first increases and then drops as the number of printers increases. Except for the case with only two printers on the aisle, zone-picking gives higher average WIC than the policy of two bucket brigades.

5.7 Effects of bucket size

Define the *bucket size* as the maximum number of pick-lists that a worker can carry at a time. We assume that each time a worker takes as many pick-lists as possible (as long as they are available) up to this number from a printer. We investigate the effects of bucket size on all the order-picking policies discussed in the previous section.

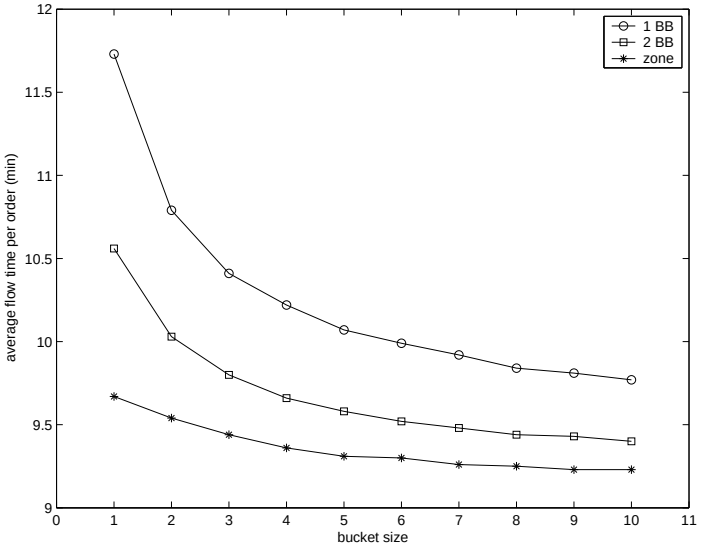

Figure 5.16: The average flow time of an order decreases with bucket size.

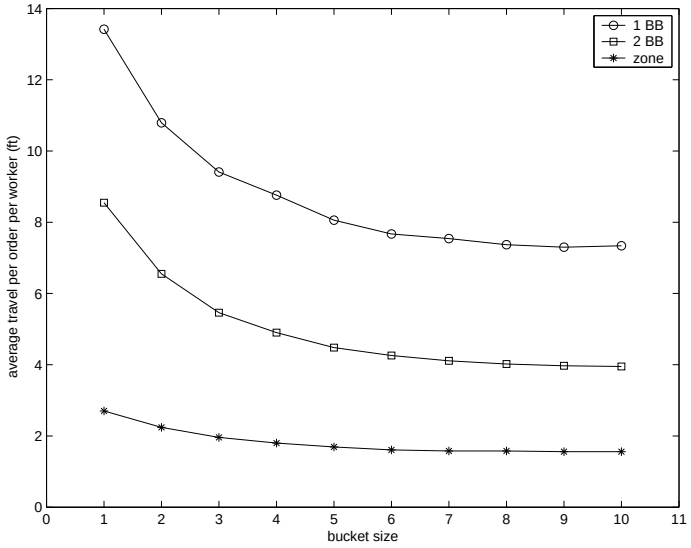

Figure 5.17: The average travel per order of a worker decreases with bucket size.

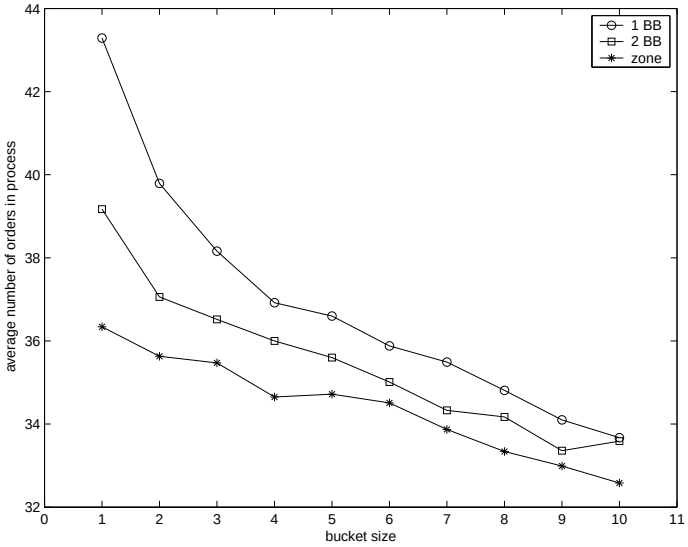

Figure 5.18: The average WIP on the aisle decreases (with some small fluctuation) with bucket size.

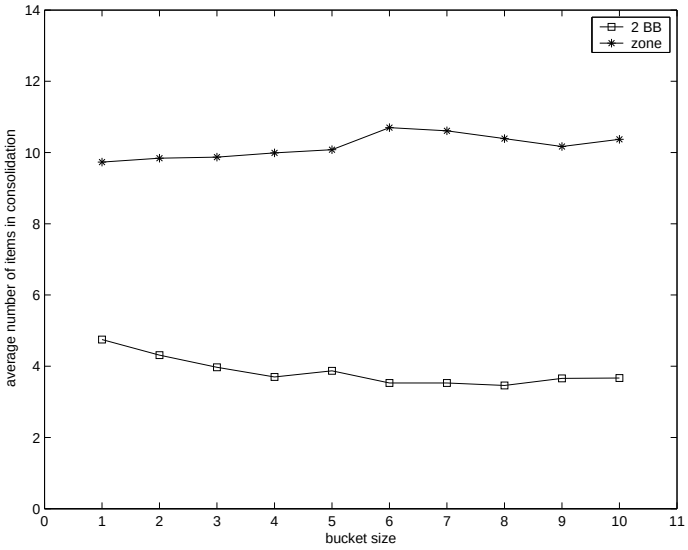

Figure 5.19: Zone-picking results in higher average WIC than the policy of two bucket brigades.

Figure 5.16 shows the average flow time per order, under different order-picking policies, with different bucket sizes. The number of printers is set at 14, which equals the number of workers on the aisle. In Figure 5.16 bucket size increases from 1 to 10. The average flow time per order decreases with bucket size. Among all policies zone-picking gives the lowest average flow time, whereas a single bucket brigade gives the highest.

Figures 5.17 and 5.18 show the average travel per order per worker (in feet) and the average WIP on the aisle (in orders) respectively, under different order-picking policies, with different bucket sizes. The same set of printer locations for Figure 5.16 is used for each order-picking policy and each bucket size. The results are similar to the average flow time. Both average travel and WIP decrease with bucket size. Among all policies zone-picking gives the lowest average travel and WIP, whereas a single bucket brigade gives the highest.

Figure 5.19 shows the average WIC (in items) by two bucket brigades and zone-picking. The average WIC remains more or less constant for both policies as the bucket size increases. Zone-picking results in higher average WIC than the policy of two bucket brigades.

5.8 What is the best policy?

Table 5.2: Zone-picking gives lower values in average flow time, average travel, and average WIP. However, it causes more orders to be split and therefore, higher average WIC. This increases the work in sortation and consolidation.

	1 bucket brigade	2 bucket brigades	Zone-picking
Average flow time per order	High	Intermediate	Low
Average travel per order per worker	High	Intermediate	Low
Average work-in-process (WIP)	High	Intermediate	Low
Average work-in-consolidation (WIC)	No WIC	Intermediate	High

The overall performance of different order-picking policies in the simulations is sum-

marized in Table 5.2. Results indicate that, independent of the number of printers and bucket size, zone-picking always gives the lowest average flow time, average travel, and average WIP. However, zone-picking requires more orders to be split and therefore, results in higher WIC than bucket brigades. This increases the work in sortation and consolidation. Effectively, it shifts the work load to downstream from order-picking by splitting each order into pick-lists (if necessary) and processing the pick-lists in parallel. On the other hand, a single bucket brigade progressively assembles each order across the entire aisle. It does not require orders to be split. This eliminates the work in sortation and consolidation. However, it gives higher average flow time, average travel, and average WIP in the order-picking operation.

Minimizing the cycle time of an order is the most important concern of the distribution center. The cycle time of an order includes the time spent in order-picking as well as the time for sortation and consolidation. Bucket brigades are "slow" in order-picking but "fast" in the sorting-consolidating stage, whereas zone-picking is "fast" in order-picking but "slow" in sortation and consolidation. Which policy should we use in order to minimize the average cycle time of an order? Let t_{s-c} denote the expected time to sort and consolidate a pick-list (and its retrieved SKUs) with other pick-lists of the same order. Let T_{zone} be the total time to sort and consolidate all the pick-lists of an order under zone-picking. For example, if an order is split into 4 pick-lists then $E[T_{zone}] = 3 \times t_{s-c}$. In general,

$$E[T_{zone}] \quad = \quad P_2 \times t_{s-c} + P_3 \times 2t_{s-c} + \ldots + P_m \times (m-1)t_{s-c};$$

where P_i is the probability of an order split into i pick-lists under zone-picking, and m is the number of printers on the aisle. Let f_{1BB} and f_{zone} be the average flow time per order by one bucket brigade and zone-picking respectively. Running a single bucket brigade is preferable to zone-picking if

$$f_{1BB} \quad < \quad f_{zone} + E[T_{zone}];$$

which is equivalent to

$$t_{s-c} \quad > \quad t^*_{1BB-zone}, \tag{5.1}$$

where $t^*_{1BB-zone} \equiv (f_{1BB} - f_{zone})/\left[P_2 + P_3 \times 2 + \ldots + P_m \times (m-1)\right]$. We call $t^*_{1BB-zone}$ the *critical sorting-consolidating time when comparing the policy of one bucket brigade with zone-picking*. Running one bucket brigade across the entire aisle is preferable to zone-picking if the expected time t_{s-c} to sort and consolidate a pick-list with other pick-lists of the same order is greater than the critical sorting-consolidating time $t^*_{1BB-zone}$.

Table 5.3: Comparison on the average flow time per order (in minutes) by a single bucket brigade and zone-picking with different bucket sizes

Bucket size	1	2	3	4	5	6	7	8	9	10
f_{1BB}	11.73	10.79	10.41	10.22	10.07	9.99	9.92	9.84	9.81	9.77
f_{zone}	9.67	9.54	9.44	9.36	9.31	9.30	9.26	9.25	9.23	9.23
$f_{1BB} - f_{zone}$	2.06	1.25	0.97	0.86	0.76	0.69	0.66	0.59	0.58	0.54
$t^*_{1BB-zone}$	5.17	3.13	2.43	2.15	1.90	1.72	1.65	1.48	1.45	1.35

Table 5.3 compares the average flow time per order under one bucket brigade and zone-picking with different bucket sizes. The number of printers is set at 14. Table 5.3 also shows the difference in average flow time per order $f_{1BB} - f_{zone}$ and the critical sorting-consolidating time $t^*_{1BB-zone}$.

Similarly, we can determine if running two bucket brigades is preferable to zone-picking. Let f_{2BB} be the average flow time per order by running two bucket brigades. Let T_{2BB} denote the total time to sort and consolidate all the pick-lists of an order under the policy of two bucket brigades. Then

$$E\left[T_{2BB}\right] \quad = \quad P'_2 \times t_{s-c},$$

where P'_2 is the probability of an order split into two pick-lists under the policy of two bucket brigades. Running two bucket brigades is preferable to zone-picking if

$$f_{2BB} + E\left[T_{2BB}\right] \quad < \quad f_{zone} + E\left[T_{zone}\right];$$

which is equivalent to

$$t_{s-c} \quad > \quad t^*_{2BB-zone}, \tag{5.2}$$

where $t^*_{2BB-zone} \equiv (f_{2BB} - f_{zone}) / [P_2 + P_3 \times 2 + \ldots + P_m \times (m-1) - P'_2]$. We call $t^*_{2BB-zone}$ the *critical sorting-consolidating time when comparing the policy of two bucket brigades with zone-picking*. Running two bucket brigades is preferable to zone-picking if the expected sorting-consolidating time t_{s-c} is greater than the critical sorting-consolidating time $t^*_{2BB-zone}$.

Table 5.4: Comparison on the average flow time per order (in minutes) by two bucket brigades and zone-picking with different bucket sizes

Bucket size	1	2	3	4	5	6	7	8	9	10
f_{2BB}	10.56	10.03	9.80	9.66	9.58	9.52	9.48	9.44	9.43	9.40
f_{zone}	9.67	9.54	9.44	9.36	9.31	9.30	9.26	9.25	9.23	9.23
$f_{2BB} - f_{zone}$	0.89	0.49	0.36	0.30	0.27	0.22	0.22	0.19	0.20	0.17
$t^*_{2BB-zone}$	3.03	1.66	1.22	1.02	0.91	0.74	0.74	0.64	0.68	0.57

Table 5.4 compares the average flow time per order under two bucket brigades and zone-picking with different bucket sizes. It also shows the difference in average flow time per order $f_{2BB} - f_{zone}$ and the critical sorting-consolidating time $t^*_{2BB-zone}$.

Finally, a single bucket brigade is preferable to two bucket brigades if

$$f_{1BB} \quad < \quad f_{2BB} + E[T_{2BB}];$$

which is equivalent to

$$t_{s-c} \quad > \quad t^*_{1BB-2BB}, \tag{5.3}$$

where $t^*_{1BB-2BB} \equiv (f_{1BB} - f_{2BB}) / P'_2$. We call $t^*_{1BB-2BB}$ the *critical sorting-consolidating time when comparing one bucket brigade with two bucket brigades*. Running one bucket brigade is preferable to two bucket brigades if the expected sorting-consolidating time t_{s-c} is greater than the critical sorting-consolidating time $t^*_{1BB-2BB}$.

Table 5.5: Comparison on the average flow time per order (in minutes) by one and two bucket brigades with different bucket sizes

Bucket size	1	2	3	4	5	6	7	8	9	10
f_{1BB}	11.73	10.79	10.41	10.22	10.07	9.99	9.92	9.84	9.81	9.77
f_{2BB}	10.56	10.03	9.80	9.66	9.58	9.52	9.48	9.44	9.43	9.40
$f_{1BB} - f_{2BB}$	1.17	0.76	0.61	0.56	0.49	0.47	0.44	0.40	0.38	0.37
$t^*_{1BB-2BB}$	11.19	7.26	5.83	5.35	4.68	4.49	4.20	3.82	3.63	3.53

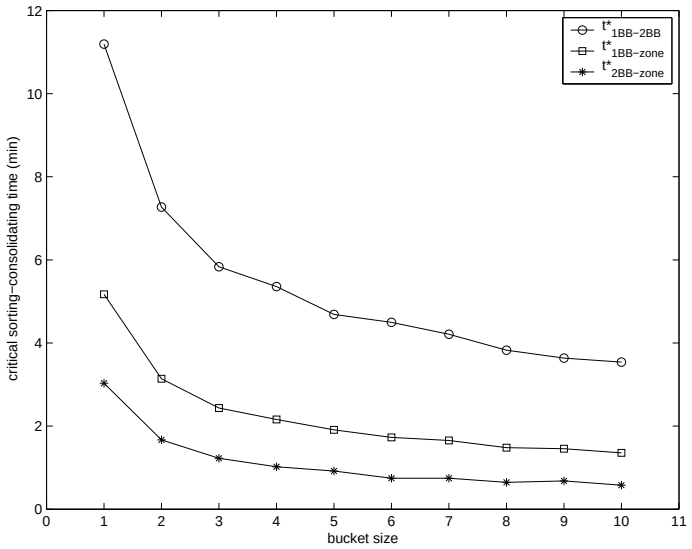

Figure 5.20: Critical sorting-consolidating times with different bucket sizes

Table 5.5 compares the average flow time per order under one and two bucket brigades with different bucket sizes. It also shows the difference in average flow time per order $f_{1BB} - f_{2BB}$ and the critical sorting-consolidating time $t^*_{1BB-2BB}$.

Figure 5.20 shows the critical sorting-consolidating times with different bucket sizes. Note that all critical sorting-consolidating times decrease with bucket size. Furthermore, for any bucket size $b \in [1, 10]$,

$$t^*_{1BB-2BB}(b) \quad > \quad t^*_{1BB-zone}(b) \quad > \quad t^*_{2BB-zone}(b). \tag{5.4}$$

Using inequalities (5.4) together with criteria (5.1), (5.2), and (5.3), an order-picking policy should be selected according to the following principle.

Principle 5.1. *Given the bucket size b and the expected sorting-consolidating time t_{s-c},*

1. *if* $t_{s-c} > t^*_{1BB-2BB}(b)$*, then the policy of one bucket brigade is superior to the policy of two bucket brigades, which in turn is superior to the policy of zone-picking;*

2. *if* $t^*_{1BB-2BB}(b) > t_{s-c} > t^*_{1BB-zone}(b)$*, then the policy of two bucket brigades is superior to the policy of one bucket brigade, which in turn is superior to the policy of zone-picking;*

3. *if* $t^*_{1BB-zone}(b) > t_{s-c} > t^*_{2BB-zone}(b)$*, then the policy of two bucket brigades is superior to the policy of zone-picking, which in turn is superior to the policy of one bucket brigade;*

4. *if* $t_{s-c} < t^*_{2BB-zone}(b)$*, then the policy of zone-picking is superior to the policy of two bucket brigades, which in turn is superior to the policy of one bucket brigade.*

5.9 Summary

Although zone-picking processes orders quickly in order-picking, it generates extra work in sortation and consolidation, which delay the departure of orders. On the other hand, a single bucket brigade progressively assembles orders and therefore, splits no orders within the aisle. However, it takes longer time to assemble orders in the order-picking operation.

Zone-picking gives the lowest average travel because the motion of each worker is restricted to a static zone. Workers do not need to walk along the entire aisle. On the other hand, due to the low pick density, workers in a single bucket brigade must walk considerable distance to pick up work from printers or hand off work to co-workers. Upon finishing their work, they must walk all the way back to their respective primary printers. All these create extra travel for a bucket brigade worker.

It could be misleading to conclude that zone-picking generates the lowest WIP on the aisle. Although it gives the lowest average WIP on the aisle among all the policies in the simulations, one should be aware that zone-picking creates more work at the sorting and

consolidating area. It simply shifts the work load to the downstream from order-picking.

Running two bucket brigades on the aisle serves as a compromise between zone-picking and a single bucket brigade. It splits fewer orders than zone-picking and gives lower average flow time than a single bucket brigade.

To minimize the average cycle time of an order, a policy should be selected according to Principle 5.1, which is based on the comparison on the sum of the average flow time and the expected sorting-consolidating time of an order under any two different policies.

Chapter 6

Conclusions

The main contribution of our work is to show how the ideas of bucket brigade assembly lines can be generalized to more complex environments. We have generalized the ideas in several ways. These include

- an adapted bucket brigade protocol for in-tree assembly networks;
- a generalized model that permits chaotic behavior, which demonstrates that the output of a purely deterministic system can be so variable that it is effectively random;
- a more detailed protocol for a flow line on which jobs arrive arbitrarily in time and are introduced into the system at multiple points along the line.

We extend the ideas of bucket brigades to in-tree assembly networks, so that even relatively complex assembly systems can enjoy the benefits of self-balancing. This can be achieved by conceptually converting the work content on an assembly network to a one-dimensional sequence of work upon which the adapted bucket brigade protocol can be executed. The system converges to a stationary state in which every worker repeats the same portion of work content (possibly across several subassembly lines) on each successive instance of the product. Furthermore, to a good approximation, the production rate of the system attains the maximum possible.

The ideas of bucket brigades are applicable even on more restrictive in-trees on which workers must travel along the subassembly lines and they are not allowed to pass each other. Due to these additional constraints, the conversion of work content on an in-tree to a one-dimensional sequence of work must be done more carefully. Furthermore, workers must begin at proper locations on the in-tree so that the system can achieve self-balance.

Another direction of our research is to model the situations in which workers are allowed to pass each other and the walk-back time to receive work is significant. We analyzed a generalized model that captures these complications. An interesting property of this generalized model is the capacity of exhibiting chaotic behavior. Even in the case with only two workers, the dynamics of the system can be chaotic and may induce large variability in completion times. This demonstrates that the variability caused by the dynamics of a purely deterministic system can be so large that the output of the system is effectively random. Furthermore, different areas in the chaotic domain give rise to different levels of variability.

We adapted the above model to the order-picking operation of a distribution center at Atlanta, where workers spend significant time to walk back to receive work and are allowed to pass each other. Furthermore, an adapted bucket brigade protocol that deals with multiple job sources along a flow line has been devised. Simulation results suggest that bucket brigades generate more work in order-picking but less work in consolidation (which occurs at the downstream from order-picking) than zone-picking because orders are progressively assembled along the flow line in a bucket brigade. Thus, for the distribution center the superiority of bucket brigades depends critically on the expected time to sort and consolidate a split order.

Appendix A

Technical Details

A.1 *Region 1* $(v_1 > v_2; w_1 \geq w_2)$

To construct the map in this region, we first establish the following lemma.

Lemma A.1. *If $x^k \in [P_r, 1]$, then there are up to r forward overtakings before the $(k+1)$st hand-off, where*

$$P_r = \frac{1/v_2 - 1/v_1 - r(1/v_1 + 1/w_1)}{1/v_2 + 1/w_1}. \tag{A.1}$$

Furthermore, the maximum such r is

$$\overline{r} = \left\lfloor \frac{1/v_2 - 1/v_1}{1/v_1 + 1/w_1} \right\rfloor. \tag{A.2}$$

Proof. In this region, only forward overtaking is possible. At the kth hand-off, worker 1 relinquishes some item i to worker 2. After the hand-off, worker 1 walks back to initiate a new item whereas worker 2 continues to work on item i. To have r forward overtakings before the next hand-off, worker 1 must first reach 0 (the beginning of the line), complete r items successively by himself, and finally meet worker 2 while worker 2 is walking back after finishing item i. Furthermore, if the kth hand-off occurs at point P_r then the next

hand-off will occur at the very end of the line. Thus we have the following relation:

$$\frac{P_r}{w_1} + r\left(\frac{1}{v_1} + \frac{1}{w_1}\right) + \frac{1}{v_1} \quad = \quad \frac{1 - P_r}{v_2}.$$

Equation (A.1) follows by solving for P_r.

$\overline{r}$ is the maximum r such that P_r remains nonnegative, that is, $P_r \geq 0$. This gives

$$r \leq \frac{1/v_2 - 1/v_1}{1/v_1 + 1/w_1}.$$

Since r is integral, the last inequality implies Equation (A.2). □

The above lemma immediately leads to the following corollary.

Corollary A.1. *The maximum number of forward overtakings in Region 1 is* $\overline{r} + 1$.

Proof. Lemma A.1 implies that there will be $\overline{r}+1$ forward overtakings before the $(k+1)$st hand-off if $x^k \in [0, P_{\overline{r}})$, and this is the maximum number of overtakings possible in this region. □

Note that the maximum number of overtakings in this region is independent of w_2. To construct the map, we need to know exactly how many overtakings occur before the $(k+1)$st hand-off given the value of x^k. The following corollary determines this.

Corollary A.2. *There are exactly* $\overline{r}+1, \overline{r}, \ldots, 1$*, and 0 forward overtakings before the* $(k+1)$*st hand-off if* x^k *falls within* $[0, P_{\overline{r}}), [P_{\overline{r}}, P_{\overline{r}-1}), \ldots, [P_1, P_0)$*, and* $[P_0, 1]$ *respectively.*

Proof. From Lemma A.1 we know that if $x^k \in [P_0, 1]$ then there is no forward overtaking before the next hand-off. Furthermore, if $x^k \in [P_1, 1]$ then there is up to one forward overtaking before the next hand-off. These imply that if $x^k \in [P_1, P_0)$ then there is exactly one forward overtaking before the next hand-off. The results for other intervals can be derived in an inductive manner. □

Using the result of Corollary A.2. We are ready to construct the map in this region.

Proof of Lemma 4.3.

According to Corollary A.2, if $x^k \in [P_r, P_{r-1}), r = \overline{r}, \overline{r}-1, \ldots, 1$, there are exactly r forward overtakings before the next hand-off. Thus, we have the following relation:

$$\frac{x^k}{w_1} + r\left(\frac{1}{v_1} + \frac{1}{w_1}\right) + \frac{x^{k+1}}{v_1} \quad = \quad \frac{1-x^k}{v_2} + \frac{1-x^{k+1}}{w_2}.$$

After rearranging terms, we have

$$x^{k+1} \quad = \quad \frac{1/v_2 + 1/w_2 - r(1/v_1 + 1/w_1)}{1/v_1 + 1/w_2} - \frac{1/v_2 + 1/w_1}{1/v_1 + 1/w_2}\, x^k.$$

The results for the intervals $[0, P_{\overline{r}})$ and $[P_0, 1]$ can be derived in a similar manner. □

A.2 *Region 2* $(v_1 \leq v_2; w_1 \geq w_2)$

Proof of Lemma 4.4.

Since overtaking is impossible in this region, after the kth hand-off worker 1 walks back to the start of the line, initiates a new item, and finally relinquishes the item to worker 2 in the next hand-off. Thus, we have the following relation:

$$\frac{x^k}{w_1} + \frac{x^{k+1}}{v_1} \quad = \quad \frac{1-x^k}{v_2} + \frac{1-x^{k+1}}{w_2}.$$

The result follows by solving for x^{k+1}. □

A.3 *Region 3* $(v_1 \leq v_2; w_1 < w_2)$

To construct the map in this region, we first establish the following lemma.

Lemma A.2. *If $x^k \in [0, P_s]$, then there are up to s backward overtakings before the (k+1)st hand-off, where*

$$P_s \quad = \quad \frac{(s+1)(1/v_2 + 1/w_2)}{1/v_2 + 1/w_1}. \tag{A.3}$$

Furthermore, the maximum such s is

$$\overline{s} \;=\; \left\lfloor \frac{1/w_1 - 1/w_2}{1/v_2 + 1/w_2} \right\rfloor. \tag{A.4}$$

Proof. In this region, only backward overtaking is possible. At the kth hand-off, worker 1 relinquishes some item i to worker 2. After the hand-off, worker 1 walks back to initiate an item j while worker 2 continues to work on item i. To have s backward overtakings before the next hand-off, worker 2 has to first finish item i, complete s items successively by himself, and finally walk back to meet worker 1 while worker 1 is working on item j. Furthermore, if the kth hand-off occurs at point P_s then the next hand-off will occur at the very start of the line. Thus, we have the following relation:

$$\frac{P_s}{w_1} \;=\; \frac{1 - P_s}{v_2} + s\left(\frac{1}{w_2} + \frac{1}{v_2}\right) + \frac{1}{w_2}.$$

Equation (A.3) follows by solving for P_s.

$\overline{s}$ is the maximum s such that $P_s \leq 1$. This gives

$$s \leq \frac{1/w_1 - 1/w_2}{1/v_2 + 1/w_2}.$$

Since s is integral, the last inequality implies Equation (A.4). □

The above lemma immediately leads to the following corollary.

Corollary A.3. *The maximum number of backward overtakings in Region 3 is $\overline{s} + 1$.*

Proof. Lemma A.2 implies that there will be $\overline{s}+1$ backward overtakings before the $(k+1)$st hand-off if $x^k \in (P_{\overline{s}}, 1]$; and this is the maximum number of backward overtakings possible in this region. □

Note that the maximum number of backward overtakings in this region is independent of v_1. To construct the map, we need to know exactly how many backward overtakings occur before the $(k+1)$st hand-off given the value of x^k. The following corollary determines this.

Corollary A.4. *There are exactly* $0, 1, \ldots, \overline{s}$, *and* $\overline{s}+1$ *backward overtakings before the* $(k+1)$*st hand-off if* x^k *falls within the interval* $[0, P_0], (P_0, P_1], \ldots, (P_{\overline{s}-1}, P_{\overline{s}}]$, *and* $(P_{\overline{s}}, 1]$ *respectively.*

Proof. From Lemma A.2 we know that if $x^k \in [0, P_0]$ then there is no overtaking before the next hand-off. Furthermore, if $x^k \in [0, P_1]$ then there is up to one backward overtaking before the next hand-off. These results imply that if $x^k \in (P_0, P_1]$ then there is exactly one backward overtaking before the next hand-off. The results for other intervals can be derived in an inductive manner. □

Using the result of Corollary A.4. We are ready to construct the map in this region.

Proof of Lemma 4.5.

According to Corollary A.4, if $x^k \in (P_{s-1}, P_s], s = 1, 2, \ldots, \overline{s}$, there are exactly s backward overtakings before the next hand-off. Thus, we have the following relation:

$$\frac{x^k}{w_1} + \frac{x^{k+1}}{v_1} \quad = \quad \frac{1-x^k}{v_2} + s\left(\frac{1}{w_2} + \frac{1}{v_2}\right) + \frac{1-x^{k+1}}{w_2}.$$

After rearranging terms, we have

$$x^{k+1} \quad = \quad (s+1)\frac{1/v_2 + 1/w_2}{1/v_1 + 1/w_2} - \frac{1/v_2 + 1/w_1}{1/v_1 + 1/w_2}\, x^k.$$

The results for the intervals $[0, P_0]$ and $(P_{\overline{s}}, 1]$ can be derived in a similar manner. □

A.4 *Region 4* $(v_1 > v_2; w_1 < w_2)$

Recall that Z^+ is the set of nonnegative integers. Furthermore, for any $x \in [0, 1]$,

$$P(x) \quad = \quad \left\{(r, s) : \frac{1}{v_2} - \beta x - \frac{1}{v_1} \leq r\tau_1 - s\tau_2 \leq \frac{1}{v_2} - \beta x + \frac{1}{w_2}, r, s \in Z^+\right\}, \quad \text{(A.5)}$$

where $\tau_1 = 1/v_1 + 1/w_1$, $\tau_2 = 1/v_2 + 1/w_2$, and $\beta = 1/v_2 + 1/w_1$.

Lemma A.3. *For any* $x \in [0, 1]$, *if* $P(x)$ *is nonempty then there exists an ordered pair* $(r^*, s^*) \in P(x)$, *which is pairwise minimal in* $P(x)$.

Proof. We prove this by contradiction. Suppose there exists no ordered pair that is pairwise minimal in $P(x)$. Then there must exist some (r', s') and (r'', s'') in $P(x)$ such that one of the following cases is satisfied:

1. $r' < r''$ and $s' \geq s''$;
2. $r' \geq r''$ and $s' < s''$;
3. $r' > r''$ and $s' \leq s''$;
4. $r' \leq r''$ and $s' > s''$.

We show that all of the above cases are invalid as follows.

1. Since $r' < r'', s' \geq s''$, and

$$\begin{aligned}(r''-1)\tau_1 - s''\tau_2 &= r''\tau_1 - s''\tau_2 - \tau_1; \\ &\leq \frac{1}{v_2} - \beta x + \frac{1}{w_2} - \tau_1; \\ &< \frac{1}{v_2} - \beta x - \frac{1}{v_1}.\end{aligned}$$

   Thus, (r', s') cannot be in $P(x)$, which contradicts the assumption.

2. Since $r' \geq r'', s' < s''$, and

$$\begin{aligned}r''\tau_1 - (s''-1)\tau_2 &= r''\tau_1 - s''\tau_2 + \tau_2; \\ &\geq \frac{1}{v_2} - \beta x - \frac{1}{v_1} + \tau_2; \\ &> \frac{1}{v_2} - \beta x + \frac{1}{w_2}.\end{aligned}$$

   Thus, (r', s') cannot be in $P(x)$, which contradicts the assumption.

3. Since $r' > r'', s' \leq s''$, and

$$\begin{aligned}(r''+1)\tau_1 - s''\tau_2 &= r''\tau_1 - s''\tau_2 + \tau_1; \\ &\geq \frac{1}{v_2} - \beta x - \frac{1}{v_1} + \tau_1; \\ &> \frac{1}{v_2} - \beta x + \frac{1}{w_2}.\end{aligned}$$

Thus, (r', s') cannot be in $P(x)$, which contradicts the assumption.

4. Since $r' \leq r'', s' > s''$, and

$$\begin{aligned} r''\tau_1 - (s''+1)\tau_2 &= r''\tau_1 - s''\tau_2 - \tau_2; \\ &\leq \frac{1}{v_2} - \beta x + \frac{1}{w_2} - \tau_2; \\ &< \frac{1}{v_2} - \beta x - \frac{1}{v_1}. \end{aligned}$$

Thus, (r', s') cannot be in $P(x)$, which contradicts the assumption.

The contradicting results in the above four cases show that there must exist an (r^*, s^*) in $P(x)$, which is pairwise smaller than all other elements in $P(x)$. This completes the proof. □

Proof of Lemma 4.6.

Immediately after the kth hand-off, worker 1 walks back and worker 2 continues on the work received from worker 1. Depending on the location of the kth hand-off and the velocities of workers, there may be forward and/or backward overtakings before the $(k+1)$st hand-off. According to Corollary 4.1 (4.2) worker 1 (2) completes his item after each forward (backward) overtaking. Thus, between these two successive hand-offs the workers work "independently". Each worker carries an item until he finishes it. He then walks back to initiate a new item, completes the new item by himself, walks back again, and so on. The motion of a worker can be viewed as independent of the other worker. This process persists until worker 1 meets worker 2 while the former is working forward and the latter is walking back (that is, the $(k+1)$st hand-off).

There are two cases that will lead to a hand-off. In the first case (case I) a hand-off will occur when worker 2 starts to walk back while worker 1 is already on his way toward the end of the line. In the second case (case II) a hand-off will occur when worker 1 begins a new item while worker 2 is already on his way back to receive work. We consider these two cases separately.

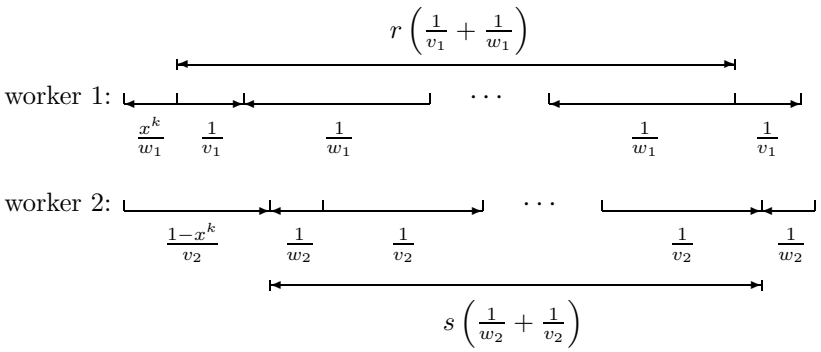

Figure A.1: The periods for a worker to work forward and walk back between the kth and $(k+1)$st hand-offs are represented by intervals with different lengths. The length of each interval represents the duration of the corresponding motion (working forward/walking back). We assume that the kth hand-off occurs at time 0. In case I, the $(k+1)$st hand-off occurs at time t, where $t \in \left[\frac{1-x^k}{v_2}+s\left(\frac{1}{w_2}+\frac{1}{v_2}\right), \frac{x^k}{w_1}+r\left(\frac{1}{v_1}+\frac{1}{w_1}\right)+\frac{1}{v_1}\right]$.

Figure A.1 illustrates case I. For each worker the periods to work forward and the periods to walk back, after the kth hand-off, are represented by intervals with different lengths in Figure A.1. An arrow is associated with each interval to indicate the direction of the corresponding motion: a right arrow represents working forward, whereas a left arrow represents walking back. The length of each interval represents the duration of the corresponding motion.

Assume that the kth hand-off occurs at time 0. From Figure A.1 it is clear that the $(k+1)$st hand-off occurs at time t, where

$$t \in \left[\frac{1-x^k}{v_2}+s\left(\frac{1}{w_2}+\frac{1}{v_2}\right), \frac{x^k}{w_1}+r\left(\frac{1}{v_1}+\frac{1}{w_1}\right)+\frac{1}{v_1}\right], \tag{A.6}$$

for some nonnegative integers r and s that satisfy

$$\frac{x^k}{w_1}+r\left(\frac{1}{v_1}+\frac{1}{w_1}\right) \le \frac{1-x^k}{v_2}+s\left(\frac{1}{w_2}+\frac{1}{v_2}\right) \le \frac{x^k}{w_1}+r\left(\frac{1}{v_1}+\frac{1}{w_1}\right)+\frac{1}{v_1};$$

which is equivalent to

$$\frac{1}{v_2}-\beta x^k-\frac{1}{v_1} \le r\tau_1-s\tau_2 \le \frac{1}{v_2}-\beta x^k. \tag{A.7}$$

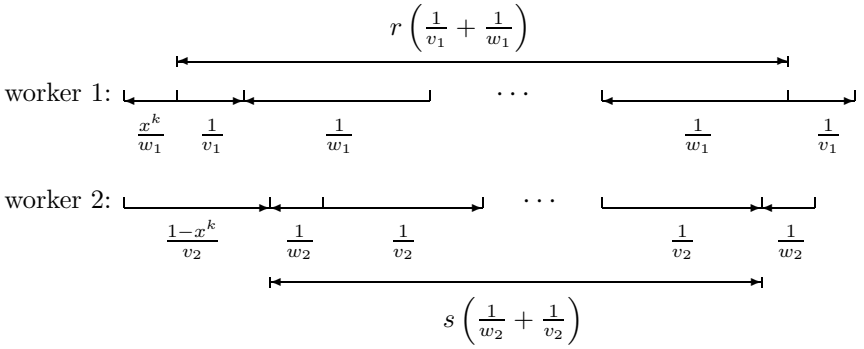

Figure A.2: Assume that the kth hand-off occurs at time 0. In case II the $(k+1)$st hand-off occurs at time $t \in \left[\frac{x^k}{w_1} + r\left(\frac{1}{v_1}+\frac{1}{w_1}\right), \frac{1-x^k}{v_2} + s\left(\frac{1}{w_2}+\frac{1}{v_2}\right) + \frac{1}{w_2}\right]$.

Figure A.2 shows another case (case II) in which the $(k+1)$st hand-off occurs at time t, where

$$t \in \left[\frac{x^k}{w_1} + r\left(\frac{1}{v_1}+\frac{1}{w_1}\right), \frac{1-x^k}{v_2} + s\left(\frac{1}{w_2}+\frac{1}{v_2}\right) + \frac{1}{w_2}\right], \tag{A.8}$$

for some nonnegative integers r and s that satisfy

$$\frac{1-x^k}{v_2} + s\left(\frac{1}{w_2}+\frac{1}{v_2}\right) \le \frac{x^k}{w_1} + r\left(\frac{1}{v_1}+\frac{1}{w_1}\right) \le \frac{1-x^k}{v_2} + s\left(\frac{1}{w_2}+\frac{1}{v_2}\right) + \frac{1}{w_2};$$

which is equivalent to

$$\frac{1}{v_2} - \beta x^k \le r\tau_1 - s\tau_2 \le \frac{1}{v_2} - \beta x^k + \frac{1}{w_2}. \tag{A.9}$$

Combining Inequalities (A.7) and (A.9), we obtain the following inequalities:

$$\frac{1}{v_2} - \beta x^k - \frac{1}{v_1} \le r\tau_1 - s\tau_2 \le \frac{1}{v_2} - \beta x^k + \frac{1}{w_2}. \tag{A.10}$$

Furthermore, each time a forward (backward) overtaking occurs r (s) is incremented by one. The $(k+1)$st hand-off occurs as soon as r and s satisfy Inequalities (A.10). That is, the $(k+1)$st hand-off occurs after r^* forward overtakings and s^* backward overtakings, where (r^*, s^*) is pairwise minimal in

$$P(x^k) = \left\{(r,s) : \frac{1}{v_2} - \beta x^k - \frac{1}{v_1} \le r\tau_1 - s\tau_2 \le \frac{1}{v_2} - \beta x^k + \frac{1}{w_2}, r, s \in Z^+\right\}.$$

Lemma A.3 guarantees that such an (r^*, s^*) pair exists.

To obtain the map in this region, we need to consider case I and case II separately. In case I, the position x^{k+1} of the $(k+1)$st hand-off satisfies the following relation

$$\frac{x^{k+1} - v_1 \triangle t}{v_1} = \frac{1 - x^{k+1}}{w_2};$$

where

$$\triangle t = \frac{1 - x^k}{v_2} + s^* \left(\frac{1}{w_2} + \frac{1}{v_2} \right) - \left[\frac{x^k}{w_1} + r^* \left(\frac{1}{v_1} + \frac{1}{w_1} \right) \right].$$

Equation (4.5) can be obtained by solving for x^{k+1}.

In case II, x^{k+1} satisfies the following relation

$$\frac{x^{k+1}}{v_1} = \frac{1 - w_2 \triangle t - x^{k+1}}{w_2};$$

where

$$\triangle t = \frac{x^k}{w_1} + r^* \left(\frac{1}{v_1} + \frac{1}{w_1} \right) - \left[\frac{1 - x^k}{v_2} + s^* \left(\frac{1}{w_2} + \frac{1}{v_2} \right) \right].$$

Similarly, Equation (4.5) can be obtained by solving for x^{k+1}. This completes the proof. □

Proof of Lemma 4.7.

We first show that the function f has only a finite number of points of discontinuity of f and f' in the interval $[0, 1]$. For $x \in [0, 1]$ denote $(r^*(x), s^*(x))$ as the (r, s) pair that is pairwise minimal in the set

$$P(x) = \left\{ (r, s) : \frac{1}{v_2} - \beta x - \frac{1}{v_1} \leq r\tau_1 - s\tau_2 \leq \frac{1}{v_2} - \beta x + \frac{1}{w_2}, r, s \in Z^+ \right\}, \qquad \text{(A.11)}$$

where $\tau_1 = 1/v_1 + 1/w_1$, $\tau_2 = 1/v_2 + 1/w_2$, and $\beta = 1/v_2 + 1/w_1$. Furthermore, define a *jump* as an event for any $x \in [0, 1]$ such that $(r^*(x), s^*(x)) \neq (r^*(x + \delta x), s^*(x + \delta x))$, for $0 < \delta x \ll 1$. A jump corresponds to a point of discontinuity of f in Equation (4.5). A jump is due to either one of the following reasons:

1. $(r^*(x), s^*(x))$ is not in $P(x+\delta x)$. A new, distinct ordered pair $(r^*(x+\delta x), s^*(x+\delta x))$ is found to be pairwise minimal in $P(x+\delta x)$.

2. $(r^*(x), s^*(x))$ is in $P(x+\delta x)$. However, there exists an emerging ordered pair $(r^*(x+\delta x), s^*(x+\delta x))$, which is pairwise smaller than $(r^*(x), s^*(x))$.

We will show that jumps due to each of the above reasons cannot occur infinitely many times.

1. Rewriting the inequalities in Equation (A.11) gives

$$s \geq -1 + \frac{\beta}{\tau_2}\, x + \frac{\tau_1}{\tau_2}\, r; \tag{A.12}$$

$$s \leq \frac{1/v_1 - 1/v_2}{\tau_2} + \frac{\beta}{\tau_2}\, x + \frac{\tau_1}{\tau_2}\, r. \tag{A.13}$$

   Given an $x \in [0,1]$, finding an ordered pair (r,s) in $P(x)$ is equivalent to finding an integer point (r,s) that lies between the real lines $y' = -1 + (\beta/\tau_2)x + (\tau_1/\tau_2)x'$ and $y' = (1/v_1 - 1/v_2)/\tau_2 + (\beta/\tau_2)x + (\tau_1/\tau_2)x'$ on the x'-y' plane. Both β and τ_2 are positive and finite because w_1, v_2, and w_2 are positive. Thus, these real lines shift upward as x increases. To have infinitely many jumps that are due to reason 1 as x increases, it is necessary that the gap between the two real lines equals 0. That is, $(1/v_1 - 1/v_2)/\tau_2 = -1 \Rightarrow v_1 = -w_2$, which is impossible.

2. For any $x \in [0,1]$ both $r^*(x)$ and $s^*(x)$ in $(r^*(x), s^*(x))$ are finite because $(r^*(x), s^*(x))$ is pairwise minimal in $P(x)$. There are only a finite number of (r,s) pairs, $r, s \in Z^+$, that are pairwise smaller than $(r^*(x), s^*(x))$. Therefore, there are only a finite number of jumps that are due to reason 2.

Combining the above two cases we conclude that we cannot have infinitely many jumps in the interval $[0,1]$. This implies that the map f has only a finite number of points of discontinuity of f and f' in the interval $[0,1]$.

Each point of discontinuity corresponds to a new (r^*, s^*) pair, which is pairwise minimal in $P(x)$, for x in some subinterval within $[0,1]$. Within this subinterval the map f

in Equation (4.5) is a linear function of x^k (because (r^*, s^*) is fixed) with slope $-\frac{\beta}{\alpha}$. This concludes that the map f in Region 4 is piecewise linear and $f' = -\frac{\beta}{\alpha}$ except for a finite number of points in $[0, 1]$. $\square$

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993. ISBN 0-136-17549-X.

[2] K.T. Alligood, T.D. Sauer, and J.A. Yorke. *Chaos: An Introduction to Dynamical Systems*. Springer, 1996. ISBN 0-387-94677-2.

[3] C. Anderson, J.J. Boomsma, and J.J. Bartholdi, III. Task partitioning in insect societies: bucket brigades. *Insectes Soc.*, 49:171–180, 2002.

[4] D. Armbruster. Dynamical systems and production systems. In G. Radons and R. Neugebauer, editors, *Nonlinear Dynamics of Production Systems*. Wiley-VCH, 2004. ISBN 3-527-40430-9.

[5] D. Armbruster and E.S. Gel. Bucket brigades revisited: Are they always effective? *Eur. J. Oper. Res.* In press.

[6] D. Armbruster, E.S. Gel, and J. Murakami. Bucket brigades with worker learning. Working paper, Department of Industrial Engineering, Arizona State University, 2004.

[7] K.R. Baker, S.G. Powell, and D.F. Pyke. Optimal allocation of work in assembly systems. *Mgmt. Sci.*, 39(1):101–106, 1993.

[8] J.J. Bartholdi, III, L.A. Bunimovich, and D.D. Eisenstein. Dynamics of two- and three-worker "bucket brigade" production lines. *Oper. Res.*, 47(3):488–491, 1999.

[9] J.J. Bartholdi, III and D.D. Eisenstein. The bucket brigade web page, `http://www.isye.gatech.edu/~jjb/bucket-brigades.html`.

[10] J.J. Bartholdi, III and D.D. Eisenstein. A production line that balances itself. *Oper. Res.*, 44(1):21–34, 1996.

[11] J.J. Bartholdi, III and D.D. Eisenstein. Chaos and convergence in bucket brigades with finite walk-balk velocities and passing. Working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2003.

[12] J.J. Bartholdi, III, D.D. Eisenstein, and R.D. Foley. Performance of bucket brigades when work is stochastic. *Oper. Res.*, 49(5):710–719, 2001.

[13] J.J. Bartholdi, III and S.T. Hackman. *Warehouse and Distribution Science*. Working text, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2003.

[14] I. Baybars. A survey of exact algorithms for the simple assembly line balancing problem. *Mgmt. Sci.*, 32(8):909–932, 1986.

[15] D.P. Bischak. Performance of a manufacturing module with moving workers. *IIE Trans.*, 28:723–733, 1996.

[16] A.I. Bratcu and A. Dolgui. A relaxation of the normative model of the self-balancing production lines ("bucket brigades"). Working paper, Centre SIMMO, École des Mines de Saint Étienne, 2004.

[17] C.J. Chase, J. Serrano, and P.J. Ramadge. Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled continuous systems. *IEEE Trans. Automat. Contr.*, 38(1):70–83, 1993.

[18] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms, 2nd Edition*. MIT Press, 2001. ISBN 0-262-03293-7.

[19] R.L. Devaney. *A First Course in Chaotic Dynamical Systems: Theory and Experiment*. Addison-Wesley Publishing Company, 1992. ISBN 0-201-55406-2.

[20] G.N. Frederickson, M.S. Hecht, and C.E. Kim. Approximation algorithms for some routing problems. *SIAM J. Comput.*, 7(2):178–193, 1978.

[21] S. B. Gershwin. *Manufacturing Systems Engineering*. Prentice Hall, 1994. ISBN 0-135-60608-X.

[22] W.J. Hopp and M.L. Spearman. *Factory Physics: Foundations of Manufacturing Management, 2nd Edition*. Irwin/McGraw-Hill, 2000. ISBN 0-256-24795-1.

[23] S.T. Hutchinson, J.R. Villalobos, and M.G. Beruvides. Effects of high labor turnover in a serial assembly environment. *Int. J. Prod. Res.*, 35(11):3201–3223, 1997.

[24] A. Lasota and M.C. Mackey. *Probabilistic Properties of Deterministic Systems*. Cambridge University Press, 1985. ISBN 0-521-30248-X.

[25] T.-Y. Li and J.A. Yorke. Ergodic transformations from an interval into itself. *Trans. Amer. Math. Soc.*, 235:183–192, 1978.

[26] A. Montano, J.R. Villalobos, L.R. Mar, and M.A. Gutierrez. Using a modified work sharing method for the reduction of labor turnover's impact on the throughput of a serial assembly line. Working paper, Department of Industrial Engineering, Arizona State University, 2002.

[27] L.F. Muñoz and J.R. Villalobos. Work allocation strategies for serial assembly lines under high labor turnover. *Int. J. Prod. Res.*, 40(8):1835–1852, 2002.

[28] K. Peters, J. Worbs, U. Parlitz, and H.-P. Wiendahl. Manufacturing systems with restricted buffer sizes. In G. Radons and R. Neugebauer, editors, *Nonlinear Dynamics of Production Systems*. Wiley-VCH, 2004. ISBN 3-527-40430-9.

[29] J.L. Reyes and F. Fernández-Haegar. Sequential co-operative load transport in the seed-harvesting ant *Messor barbarus*. *Insectes Soc.*, 46:199–125, 1999.

[30] A. Scholl. *Balancing and Sequencing of Assembly Lines, 2nd Edition*. Physica-Verlag, 1999. ISBN 3-790-81180-7.

[31] B.J. Schroer, J. Wang, and M.C. Ziemke. A look at TSS through simulation. *Bobbin*, July:114–119, 1991.

[32] E. Zavadlav, J.O. McClain, and L.J. Thomas. Self-buffering, self-balancing, self-flushing production lines. *Mgmt. Sci.*, 42(8):1151–1164, 1996.

Vita

Yun Fong Lim was born in 1973 at Batu Pahat, Johor, Malaysia. He received his B.S. in Physics from the National Central University in 1996 and his M.S. in Computational Science from the National University of Singapore in 1998. He then worked as an Engineer at Kent Ridge Digital Labs in Singapore. In August 2000 he went to Georgia Institute of Technology, where he pursues his Ph.D. studies in Industrial and Systems Engineering. His research centers on modeling and analysis of self-organizing production systems. He is also interested in using concepts and techniques from the theory of dynamical systems to analyze problems in manufacturing, services, and distribution. After obtaining his Ph.D. degree, he will join the Singapore Management University as an Assistant Professor of Operations Management.